National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

University of Alberta

Variable-Resolution Techniques for Boundary Detection and Character Thinning

by

Manoj Kumar Jain

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Fall 1992

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-77095-3

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Manoj Kumar Jain
TITLE OF THESIS: Variable-Resolution Techniques for Boundary Detection and Character Thinning

DEGREE: Master of Science
YEAR THIS DEGREE GRANTED: 1992

Permission is hereby granted to UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis not extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) . . Manoj . . . . . .

Permanent Address:
39 Parasnath St.,
Muzaffarnagar City
India 251 002

Date: 28 Aug 1992

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **Variable-Resolution Techniques for Boundary Detection and Character Thinning** submitted by **Manoj Kumar Jain** in partial fulfillment of the requirements for the degree of Master of Science.

A. Basu (Co-Supervisor)

X. Li (Co-Supervisor)

H. Zhang

Z. Koles (Applied Sciences in Medicine)

P. Gburzynski

Date: 28 Aug 1992

Abstract

Boundary detection and Character thinning are the two most important operations in the field of Computer Vision. The purpose of boundary detection algorithms is to give meaningful shapes to the relatively featureless edge image generated as a result of applying some standard edge detection technique. Similarly, the objective of the thinning algorithms is to generate the skeleton of a character one stroke thick which can be later used for the purpose of recognition. Most Computer Vision algorithms in the literature have relied on using either uniform resolution or multi-resolution techniques to extract information out of a picture. The question answered in this thesis is: how can varying resolution spatially within an image help in character thinning and boundary detection? For these problems variable resolution (VR) masks are designed, whose centers are windows in normal resolution, with each of the peripheral cells used to keep some information at a reduced resolution. Thus VR approaches effectively look at a large region of the original image, at a cost only slightly higher than processing a small region using uniform resolution schemes. The proposed algorithms are demonstrated to perform better than various well-known methods. The VR procedures described here are inherently parallel in nature. They can also be efficiently implemented on a serial computer.

Acknowledgements

I would like to thank my supervisors, Dr. Xiaobo Li and Dr. Anup Basu, for their valuable advice, inspiration and direction to this thesis. Thanks are due to the thesis examining committee members, Dr. Zoley Koles, Dr. Hong Zhang, and Dr. Pawel Gburzynski for their valuable time and suggestions. Thanks also to Mr. Steve Sutphen, Mr. Bruce Folliot, and Ms. Carol Smith for their help in preparing this thesis.

I wish to express my appreciation to my family for their patience, support, and encouragement. Finally, I would like to thank all my friends in the Computing Science department for their helpful and not so helpful suggestions.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Variable-resolution Vision 3

1.3 Organization of Thesis 4

2 Previous work **6**

2.1 Boundary Detection Algorithms 6

2.1.1 Dynamic Programming 7

2.1.2 Graph Searching Methods 9

2.1.3 Hough Transform 11

2.1.4 Contour Following 14

2.2 Thinning Algorithms 17

3 Edge Detection and Thresholding **20**

3.1 Overview 20

3.2 Edge Detection Methods 23

3.2.1 Gradient Operators 23
3.2.2 Compass Operators 26
3.2.3 Laplacian Operators and Zero Crossings 27
3.2.4 Marr and Hildreth Operator 28
3.3 Thresholding 28

4 Boundary detection **31**
4.1 Limitations of previous approaches 31
4.2 Variable-resolution masks 32
4.2.1 VR Masks for Boundary Detection 34
4.3 A VR approach to boundary following 36
4.3.1 Complexity of the Algorithm 42
4.4 Experimental results 44

5 Character thinning **59**
5.1 Limitations of small templates 59
5.2 Shape preservation using VR scheme 63
5.3 Complexity analysis of sequential method 69
5.4 Experimental results 71

6 Conclusion **84**
6.1 Contribution 84
6.2 Directions for Future Work 85

7 Bibliography 87

A Masks and Rule Windows for Boundary Detection 92

B Some more results on Character Thinning 96

List of Figures

1 Linkage Rule in Graph Search Technique 10

2 The Hough transform 12

3 Different types of Edges 22

4 Common Gradient Operators 25

5 A variable resolution template 33

6 Rasterization of Straight Lines 35

7 Mask for 22.5° orientation line 36

8 (a) a 3 × 3 template (b) a 9 × 9 template 40

9 Example Templates for detecting a vertical edge 41

10 From the top left corner in clockwise direction: Original Image of a cup, Edge Image, Image after Thresholding, Result of VR algorithm 45

11 Results of Williams' method (left side) and Lacroix's method (right side) 46

12 Test Image with no noise. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method. 48

13 Test Image with Gaussian Noise of $\mu = 0$, $\eta = 8$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method. 49

14 Test Image with Gaussian Noise of $\mu = 0$, $\eta = 32$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method. 50

15 Test Image with Gaussian Noise of $\mu = 0$, $\eta = 4$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method. 52

16 Test Image with Gaussian Noise of $\mu = 0$, $\eta = 32$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method. 53

17 From the top left corner in clockwise direction: Original Image of the cup , Williams' method, Lacroix's method, VR method 54

18 From the top left corner in clockwise direction: Original Image of the book, Williams' method, Lacroix's method, VR method 55

19 From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method . 56

20 From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method . 57

21 From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method . 58

22 Character "B" and some thinning results using Guo's algorithm A1 (a) input, (b) neighborhood of pixel "Z", (c) after iteration 5, (d) neighborhood of pixel "V", (e) the final skeleton 60

23 Character "B" and some thinning results using Holt's algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 2, (d) neighborhood of pixel "W", (e) the final skeleton . 61

24 Example templates (a) a 3×3 template, (b) and (c) are variable resolution templates. 64

25 A variable resolution template 65

26 Character "B" and some thinning results using the proposed algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 5, (d) neighborhood of pixel "T", (e) the final skeleton 66

27 Character "T" and some thinning results using the proposed algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 3, (d) neighborhood of pixel "H", (e) the final skeleton 67

28 Thinning results of the proposed algorithm for lines of various orientations 68

29 Character "D" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 72
30 Character "R" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 73
31 Character "I" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 74
32 Character "T" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 75
33 Character "N" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 75
34 Character "Z" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 76
35 Character "E" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 77
36 Character "H" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 78
37 Character "G" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 78
38 Character "Q" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 79
39 Input Gray Scale Image 80
40 Input Gray Scale Image After Thresholding 80

41 Output of Guo's Thinning Algorithm 81
42 Output of Holt's Thinning Algorithm 82
43 Output of proposed Thinning Algorithm 83
44 Character "A" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 97
45 Character "C" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 97
46 Character "F" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 98
47 Character "J" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 98
48 Character "L" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 99
49 Character "M" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method 99
50 Character "O" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method100
51 Character "P" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method100
52 Character "S" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method101

53 Character "U" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method101
54 Character "V" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method102
55 Character "W" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method102
56 Character "X" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method103
57 Character "Y" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method103

Chapter 1

Introduction

1.1 Motivation

The detection of object contours is one of the most important operations in the fields of Computer Vision and Pattern Recognition. A common approach, in the field of pattern recognition, makes use of contours to characterize and recognize objects represented in images. A contour or boundary is defined as the set of connected pixels which gives the outline of an object in a gray scale image. Boundary detection generally consists of two steps: (i) to detect all the edge elements in images and (ii) to link edge elements together to form meaningful boundary lines. The problem of detecting closed boundaries has been extensively dealt with in the literature, reflecting the importance attached to the use of contours as image features and the difficulty of achieving

an optimal method for boundary detection. Due to the presence of noise in the image, the task of locating a meaningful boundary becomes non-trivial. For example, the edge elements can exist in the absence of any boundary and may be absent at places where the boundary should be. These missing or extra edges make the task of boundary detection still more complicated.

Character thinning or skeletonization constitutes the first step in character recognition. When a written or printed character is digitized an image of the character several pixels thick is generated. The objective of thinning is to eliminate the redundant information by the successive removal of the outer layers and generate the skeleton of the character. The result of this process, during which connectivity should be preserved, is a skeleton of unit width.

Existing computer vision algorithms rely on uniform resolution and multi-resolution techniques. They use a small window (usually 3×3 template) due to the heavy computational load attached with large templates. However, small templates fail to provide global information about the shape of the character being thinned. This lack of information results in shape distortion in a thinned character. An ideal thinning algorithm should preserve the shape of the character so that misrecognition will not happen. A similar problem arises in the case of boundary extraction algorithms. One important concept has generally been overlooked: a variable resolution imaging system such as human eye can simplify visual tasks [4]. The human visual system has

a high resolution fovea and a low resolution periphery. The central viewing window allows detailed identification of objects, and the peripheral low resolution field allows fast processing. Is it possible to design thinning/boundary following algorithms which emulate this process? This problem is addressed in the thesis. The variable-resolution approaches to character thinning and boundary detection are proposed in this thesis. It is shown that the proposed methods improves upon the existing techniques.

1.2 Variable-resolution Vision

The human visual system has a high resolution sampling area and a wide visual field. The distribution of the photosensitive elements in the human retina is such that the highest resolution is limited to the central part of the sensor (fovea) and decreases towards the periphery. The role of foveal vision is to probe the environment at high resolution and, at the same time, to limit the amount of detailed information. Thus foveal vision concentrates on the objects of interest and keep intricate details about them whereas the peripheral vision provides a broad field of view [43].

Even though anthropomorphic visual systems and sensors have been fairly well studied and understood [43, 48, 44, 42, 53] their application to various vision problems still needs to be investigated. A fundamental problem with the use of such a system is the reduction of information in the periphery. In

human vision, the structure of the visual sensor and the coordinated, task-driven, active control of eye movements have an essential role in the reduction of visual information [43]. The exploitation of this structure in the context of different tasks can be partly achieved by using variable resolution templates. An extremely simplified version of variable resolution is considered. Specifically, we consider templates having two levels of resolution. A normal resolution 3×3 center, and a low resolution periphery of 8 cells, each of which represents a 3×3 region in normal resolution. The peripheral cells are used to take a "rough" view of a large neighborhood (a 9×9 window). This enables us to develop parallel algorithms which are also shape-preserving.

1.3 Organization of Thesis

The remaining portion of this thesis is organized as follows: Chapter 2 describes previous work in boundary detection and character thinning. Several existing techniques for extraction of contours are discussed with their relative advantages and disadvantages. Similarly, a brief overview of the existing thinning algorithms is given. Chapter 3 describes the principles behind edge extraction and thresholding. It briefly reviews the existing techniques and analyze them. Chapters 4 and 5 outline how VR techniques can be used to develop better boundary detection and thinning algorithms. In Chapter 4, first the basic concept behind the VR technique for extraction of boundaries

in the image is described, and then results of the method are compared with the two other existing algorithms. Chapter 5 describes the VR thinning and compares the results with two other well known approaches. Finally, Chapter 6 gives the conclusions of the work and directions for future research.

Chapter 2

Previous work

In this chapter we first describe previous work in the area of boundary detection, then we review existing literature on character thinning.

2.1 Boundary Detection Algorithms

Several boundary detection algorithms have been proposed by previous researchers. These methods, inherently sequential in nature, can be broadly classified into two categories based on the amount of knowledge incorporated into it: algorithms which incorporate *a priori* knowledge in order to form closed boundaries of the objects in images and techniques which do not rely on prior information. The term *a priori* knowledge means implicit or explicit constraints on the likelihood of a given grouping [3].

Most of the boundary following algorithms have two objectives: to measure the degree of "edgeness" at a particular image point, and to link the groups of points to obtain closed contours. For clean images most of the existing algorithms are able to obtain closed boundaries of the objects in the image. However when the image is noisy, the task of locating a good boundary becomes difficult, and often disconnected contours are obtained. Many techniques proposed in the literature use sequential segmentation which attempts to link together all the single contour elements to form either the whole object outline or a significant part of it. To achieve this, the algorithms use some amount of prior knowledge that maps the edge elements into meaningful boundaries. Ashkar and Modestino [2] list four approaches of incorporating *a priori* knowledge in sequential segmentation algorithms. These approaches are: exhaustive search, dynamic programming, structured tree search, and heuristics graph search.

2.1.1 Dynamic Programming

Dynamic Programming is a method of finding the global optimum of multistage processes. It is based on Bellman's *principle of optimality* [5], which states that *the optimum path between two given points is also optimum between any two points lying on the path.* It also reduces the computational load of exhaustive search. Montanari [35] first proposed the use of dynamic

programming to perform edge detection. He proposed a method which embeds the properties of a curve in a figure of merit representing the heuristic information. This figure of merit was then used to find the relative value of different paths. The figure of merit was then used to find the relative value of different paths. The figure of merit of a path $x_1, \ldots, x_n$ was defined as:

$$h(x_1, \ldots, x_n) = \sum_{i=1}^{n} s(x_i) + \alpha \sum_{i=1}^{n} \phi(x_i, x_{i+1})$$

where $s(x_i)$ is the edge strength at point x_i in the image and $\phi(x_i, x_{i+1})$ is the slope between the adjacent points x_i and x_{i+1}. In the above equation consecutive x_i's must be grid neighbors. This figure of merit gave the best path once all of them have been enumerated. The idea is to select the path which is a weighted sum of high cumulative edge strength and low cumulative curvature [3]. A dynamic programming technique is then used to determine the optimal curve in the image with respect to the given figure of merit. Since the figure of merit function does not guide the search, the computation time is relatively independent of the noise level in the image. This is a drawback of the method, since an ideal method should vary its computation time depending on the level of noise present in the image. The dynamic programming technique was applied by Chien and Fu [9] to find boundaries of lungs in chest X-ray films.

2.1.2 Graph Searching Methods

A boundary can also be viewed as a path through a graph formed by linking the edge elements together. Linkage rules give the procedure for connecting the edge elements [22]. In the exhaustive search technique, every contour in the image is examined and the best candidate is selected. The main drawbacks of this method are the extensive computation time and storage required. This algorithm can be applied only to images of low dimensions and is rarely used. Also, the dynamic programming approach is favored over exhaustive search technique as it reduces the computational load attached to exhaustive search.

Martelli [33, 34] formulated the problem of minimizing the figure of merit as a heuristic search for the shortest path in a graph. The graph search method needs considerably less time compared to dynamic programming. In addition, the time taken varies with the level of noise present in the image. In this technique, the candidate boundary elements are represented as graph nodes while the two neighboring boundary elements in the image defines the directed arc of the graph. The direction of the arc is obtained with the convention of moving clockwise around the first candidate boundary element. The contour is then started at an arbitrary point which qualifies as a boundary point. Thus for an edge to exist from x_i to x_j, x_j must be one of the three possible eight neighbors in front of the edge direction $\phi(x_i)$ and edge

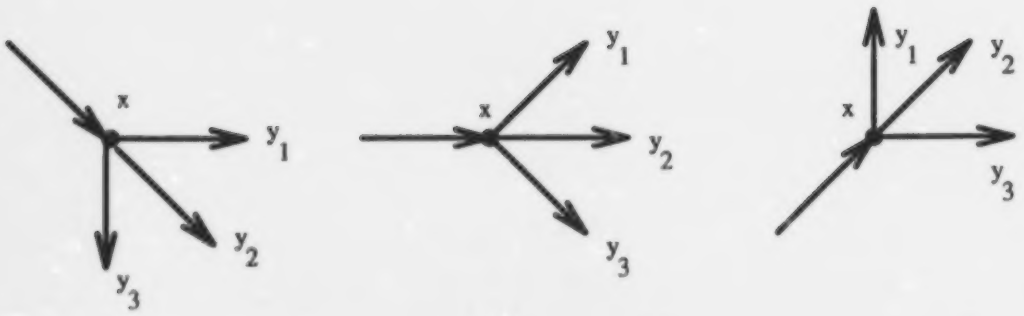


Figure 1: Linkage Rule in Graph Search Technique

strength $s(x_i) > T$ and $s(x_j) > T$ where T is the chosen threshold and $\{[\phi(x_i) - \phi(x_j)] \pmod{2\pi}\} < \frac{\pi}{2}$. For example, in Figure 1 a pixel x is considered to be linked to y if the latter is one of the three eight-connected neighbors (y_1, y_2, y_3) in front of the contour direction.

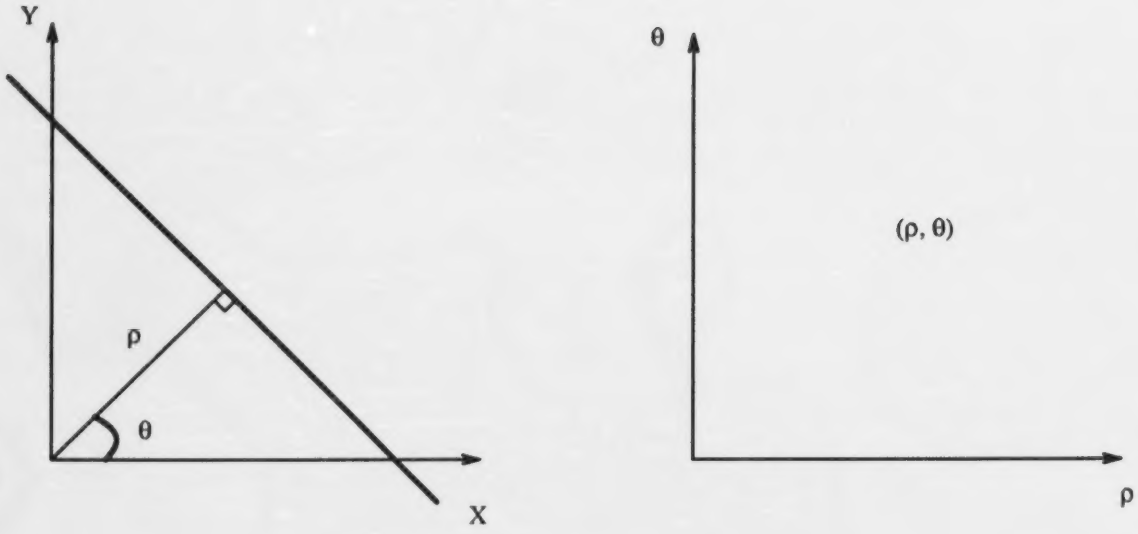
Thus a boundary is a path in the graph representing the state space and the problem of finding the best boundary reduces to finding the optimal path in the graph. The optimal path can be found by using the heuristic search technique. In heuristic search methods, an evaluation function which incorporates the properties of edges is used and the path which minimizes this function is taken as the boundary. For example, let A and B be the start and goal node in a graph with node locations x_i, $i = 1, 2, \ldots$ and $\delta(x_i)$ be the evaluation function which gives the value of the path from A to B constrained to go through the node x_i. First the successors of the start node are examined and the node which which minimizes $\delta(x_i)$ is selected. The selected node now becomes the new start node and the process is repeated

until B is reached. The sequence of selected nodes then constitutes the boundary path.

Though this procedure reduces the computation time, it is inherently sequential in nature. Another difficulty is the selection of a good evaluation function. Ashkar and Modestino [2] incorporated the idea of negative costs in the evaluation function to obtain good boundaries. The contour extraction problem was formulated as one of minimum cost tree searching. Branch costs incorporated both global and contextual information obtained from the heuristics of the problem under consideration. These costs were indicative of the likelihood that a particular branch was located on the true contour [2]. A minimum cost tree search strategy based on the Zigangirov–Jelinek stack algorithm [55, 23] was then used to extract the contours.

2.1.3 Hough Transform

Another technique which is widely used to find boundaries in images, is the Hough transform [21]. This technique for curve detection is applicable if the shape of the object in the image is known and can be described as a parametric curve. Duda and Hart [13] used Hough transforms to detect straight lines and curves. This method involves the transformation of a curve in Cartesian coordinate space to a point in polar coordination space. Consider the detection of straight lines in an image. A straight line at a



(a) Straight line in cartesian coordinate space (b) Hough transform

Figure 2: The Hough transform

distance ρ from the origin and orientation θ can be represented as

$$\rho = x \cos\theta + y \sin\theta$$

Using Hough transform this line can be defined by a single point (ρ, θ) in the $\rho - \theta$ plane; that is all the points on this line map into a single point. Figure 2(a) shows a straight line in Cartesian coordinate space and Figure 2(b) shows the same line after Hough transform in $\rho - \theta$ plane. To detect straight lines in a given set of boundary points, a parameter space is defined which gives the points that lie on the line. In this parameter space, an accumulator array $A[\rho, \theta]$ initialized to 0 for all values of ρ and θ is defined. Now $\forall(x, y)$

where it is supposed that (x, y) belongs to an edge, the members of $A[\rho, \theta]$ for which (ρ, θ) is a solution of the above equation are incremented. The local maxima in the accumulator array now correspond to the points that lie on the straight line in the image. The value of the accumulator array gives the number of points on the line. A detailed implementation of this method is given in [13]. This method can be easily generalized to detect arbitrary parametric curves. For example, to detect a circle, the edge points are transformed into the parameter space (a, b) where the equation of the circle in $x - y$ plane is

$$(x - a)^2 + (y - b)^2 = r^2$$

The main weakness of Hough transform based methods is that the computational cost and the size of accumulator array increases exponentially with the number of parameters. The Hough transform is relatively unaffected by the noise present in the image, and connects the edge elements using line and curve fitting techniques. This technique has been used successfully in several medical applications which involve radiographs. Kimme et al. [24], Wechsler and Sklansky [49], and Lantz et al. [3] used Hough transform to extract boundaries in a wide variety of applications, such as in the detection of tumors and ribs in chest radiographs.

2.1.4 Contour Following

If there is no prior knowledge about the shape of the object in the image then boundaries of the object can be recovered by using one of the simplest contour following operations. Boundary following techniques are generally sequential in nature and need a starting point. Here the image is scanned until a pixel believed to be on the boundary is encountered. This pixel is taken as the starting point of the contour. To locate the next boundary point the eight-connected neighbors are examined in the counter clockwise direction. This approach usually finds the curves in the image one at a time. A detailed survey of the boundary following algorithms can be found in [41].

Several researchers [25, 27] have used the contour following algorithm to detect boundaries. Contour following algorithms for gray-level image use edge strength and edge direction information to compute the next boundary point. Lacroix [27] used a three module strategy extract the boundaries of the objects in the images. The first module uses a conventional edge detector to compute the edge strength and orientation. The gradient direction, so computed, is digitized to point to one of the eight nearest neighbors. The second module assigns a likelihood of being an edge (LBE) to each pixel. The corresponding process is a generalization of nonmaximum deletion algorithm [6] in the sense that pixels with LBE = 0 are deleted, pixels with LBE = 1 are definitely kept as edge elements, while pixels with $0 < \text{LBE} < 1$ have

to wait for contextual information to know their status. The last module is based on Kunt's contour following algorithm [25]. In this module, a decision using contextual information is made regarding the deletion of pixels with $0 < \text{LBE} < 1$. A contour is started where the pixels have $\text{LBE} = 1$ and is continued along the edge direction as long as the LBE is < 0. A test on the length of contours so obtained is performed to remove short meaningless contours. The algorithm performs satisfactorily for less noisy images, but tends to give erroneous results if the image is contaminated with noise.

Classical contour following techniques use a search mechanism which proceed along the best path along each step and never reverse a decision once made. An alternative to this is to allow backup, that is, if an acceptance decision seems to be leading to a poor subsequent acceptance, one can go back and alter the decision. Liu [30] proposed a boundary detection algorithm with a feedback loop for backtracking. The feedback loop was incorporated into the algorithm so that the results could be checked and refined using feedback whenever necessary. Backtracking is used whenever uncertainty arises in locating the next boundary pixel. This procedure enables the algorithm to recover from errors caused by noise, but the method failed to give good results when the quality of images was poor. Chen and Siy [7] improved this algorithm by using feedback to locate the noisy areas of the image, which are smoothed to remove irregularities. The feedback mechanism is activated only when the contours obtained by the conventional algorithm are

not closed. The process continues until all the contours obtained are closed. The algorithm, tested on a 60×64 chromosome image, gave satisfactory results.

Williams and Shah [51] used the concept of scale space [52] to obtain contours. The choice of scale to use in smoothing an image has been studied in detail. Smaller scales result in too much noise and fine texture while larger scales result in delocalization of edges and gaps. In scale space, zero crossings of the second derivative are examined for a continuous spectrum of scales rather than a few discrete values. A detailed analysis of scale space can be found in [11, 45]. Here, the image is first convolved with a gradient of Gaussian operator given by

$$g(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

The set of all possible edge points are then placed on a priority queue with the edge point having largest magnitude on the top. A weight is assigned to each edge pixel based on measure of noise, a measure of curvature, contour length and gradient magnitude. The edge point with the largest magnitude is retrieved from the queue and it becomes the current boundary point. Next the direction of next edge point is computed and the point in the computed direction is examined first and then those in the adjacent directions on either side of it. The point with the maximum weight is selected as the next edge point. The weights are designed to favor the longest, strongest, and the

straightest path. The search terminates when there is no potential point left to be incorporated in the contour. Once this search is over, a similar search is started in the opposite direction and the result of two searches is combined to form a closed contour. Contours are first detected at higher scales. If a closed contour is not obtained by this process then the next finer scale is chosen and the procedure repeated until the contour cannot be extended at the finest scale. The use of finer scales produces improved detection of weak edges and helps in obtaining closed boundaries.

All the above techniques are inherently sequential in nature and require extensive computational time. A non-sequential technique for determining the boundaries is described in [39, 18]. The method incorporates knowledge in terms of a curve representing a typical shape (model) of the contour. The final contour is obtained by minimizing a function, radial inertia, defined over the gradient of the object image. Though the method is able to obtain closed contours, it requires a good estimation of parameters like centroid and orientation of the model. Errors in these parameters result in poor solutions.

2.2 Thinning Algorithms

Previous research on character thinning can be broadly classified into two categories. One group of methods [19] performs a series of tests on the neighborhood of a given pixel in order to decide whether or not the pixel

should be deleted from the character. Other schemes [10] match a set of templates with the pixel values in a given window. If a match is obtained the central pixel is removed from the binary outline.

Many thinning algorithms have previously been proposed [1, 8, 10, 14, 17, 26, 36, 54]. Surveys of various approaches can be found in [46, 38]. Guo and Hall [19] proposed two thinning algorithms, A1 and A2. Algorithm A2 is suitable for parallel processing, but A1 generates more desirable results. Theoretical proofs and extensive experimental results are given in [19] for these methods. Comparisons with several well-known thinning algorithms favor algorithm A1 over the other methods. Suen and Plamondon [38] conducted an intensive study on human thinning behavior. The character skeletons generated by human operators were used as criteria to judge machine thinning algorithms. However, it is still unclear how one can make machines simulate this human behavior.

In order to retain necessary information for correct recognition, it is important for the skeleton to preserve the shape of the original input character. For algorithm A1, and most other thinning methods, a 3×3 window is used, and the programs do not make use of information outside this window. In many situations, the lack of global knowledge about the entire character causes severe shape distortion in the resulting skeleton and leads to misclassification of the character. This point will be demonstrated in detail in Chapter 5.

The human thinning operator tends to preserve shape. For example, we do not have any difficulty distinguishing between the character "B" and the digit "8". Strictly speaking, humans "cheat" in the thinning process. We have global knowledge about the character and some knowledge about what the character and its skeleton "should" look like. On the other hand a computer program has no knowledge whatsoever outside a small (say, 3×3) window. Some thinning algorithms [17, 50] use templates larger than 3×3 to obtain knowledge of a larger neighborhood of the character pixel currently being considered, which significantly reduces the skeleton shape distortion. However, the problem with using large uniform resolution templates lies in the high computational complexity involved in the process.

Chapter 3

Edge Detection and Thresholding

3.1 Overview

Edges are image attributes which are useful for image analysis and classification in a wide range of applications. Usually edge extraction is the first step in boundary detection. One might expect that algorithms could be designed that find the boundaries of objects directly from the gray-level values in the image. This task is difficult when the boundaries have complicated shapes. A good strategy is to first transform the image into an intermediate image of local gray level discontinuities, or edges, and then connect these into meaningful boundaries.

An edge refers to places in the image where there is an abrupt change in gray level or in texture, indicating the end of one region and the beginning of another. The cross section of an ideal step edge and a more realistic representation of it is shown in Figures 3(a) and 3(b). However step edges are not the only kind of edge. Depending on the class of picture being analyzed, a variety of edge cross-sections are obtained. For example, if we look at solid objects, which contain surfaces at different orientations, meeting at sharp angles, then roof type edges and spike edges are also present [12]. For example, if the brightness values in an image increase steadily and then after a certain point decrease steadily, an edge exists at the point of change from increasing to decreasing brightness values. Such edges are called *roof edges*. The cross-section of a roof edge is shown in Figure 3(c). In the rest of the thesis, the word "edge" will refer to the step edge since they are by far the most common type of edges encountered.

Detection of edges usually involves two steps after an optional preprocessing. First edge strength and direction is assigned to each pixel. The strength may be thresholded to remove the weak pixels. Next the pixels are selected and combined into edges. These two steps are considered independently. The rest of this chapter is organized as follows: Section 3.2 reviews some of the existing edge detection algorithms. Section 3.3 gives the thresholding techniques used to remove the noise from the edge image.

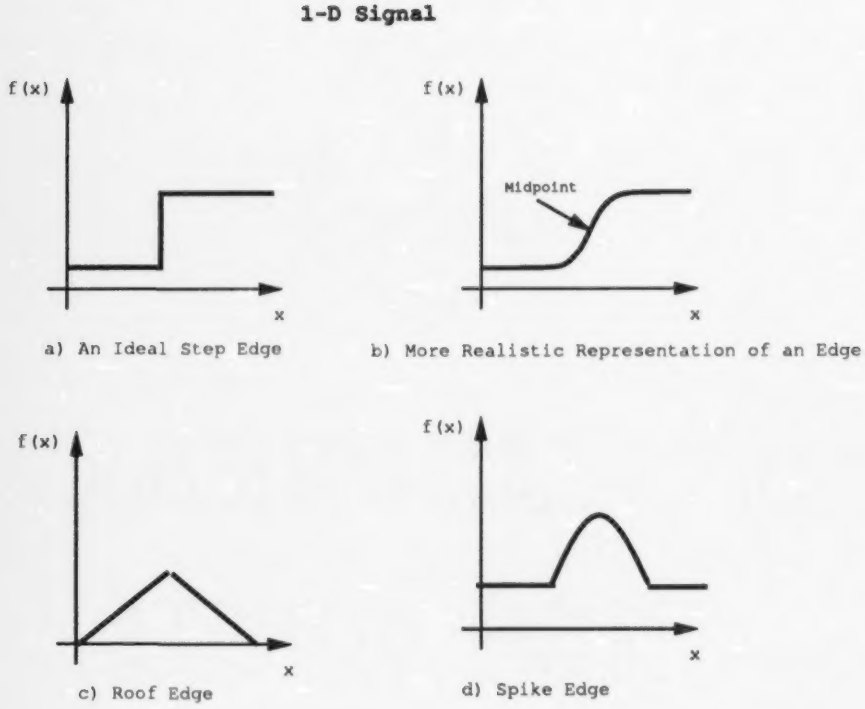


Figure 3: Different types of Edges

3.2 Edge Detection Methods

The existing edge extraction operators can be broadly classified into three categories; *gradient* operators, *compass* operators and *laplacian* operators. A detailed survey of the various operators can be found in [28, 12, 37, 41, 29]. In this section we will briefly review the most commonly used edge detectors.

3.2.1 Gradient Operators

Gradient operators are the first-order derivative operators. The gradient for a continuous two dimensional function is defined by

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

The magnitude and orientation of the gradient vector are given by the following equations

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\phi = \tan^{-1}\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$$

For a digital image, the partial derivatives can be approximated with finite differences along the two orthogonal directions x and y, thus the gradient for discrete case can be computed as

$$\nabla_x f(x,y) = f(x,y) - f(x-n,y)$$

$$\nabla_y f(x,y) = f(x,y) - f(x,y-n)$$

where n is the span of the gradient and is a small integer, usually unity. Thus $\nabla f(x,y)$ for orientation ϕ is given by

$$\nabla f(x,y) = \left(\nabla_x f(x,y)\cos\phi \, , \, \nabla_y f(x,y)\sin\phi \right)$$

and its magnitude is

$$|\nabla f(x,y)| = \sqrt{\nabla_x f(x,y)^2 + \nabla_y f(x,y)^2}$$

Some commonly used gradient operators are the Roberts operator [3], Prewitt operator [40], and Sobel operator [12]. The Roberts operator computes the finite differences about an ideal edge located at $(x + \frac{1}{2}, y + \frac{1}{2})$. Thus the finite differences are calculated diagonally instead of in orthogonal directions x and y with respect to a particular pixel. The Prewitt and Sobel operators use a 3×3 mask to approximate the gradient. These two operators compute horizontal and vertical differences of local sums and thereby reduce the effect of noise. The Sobel operator uses weights in the summation of the values of the elements to give a smoothing effect. Figure 4 shows the various gradient difference operators.

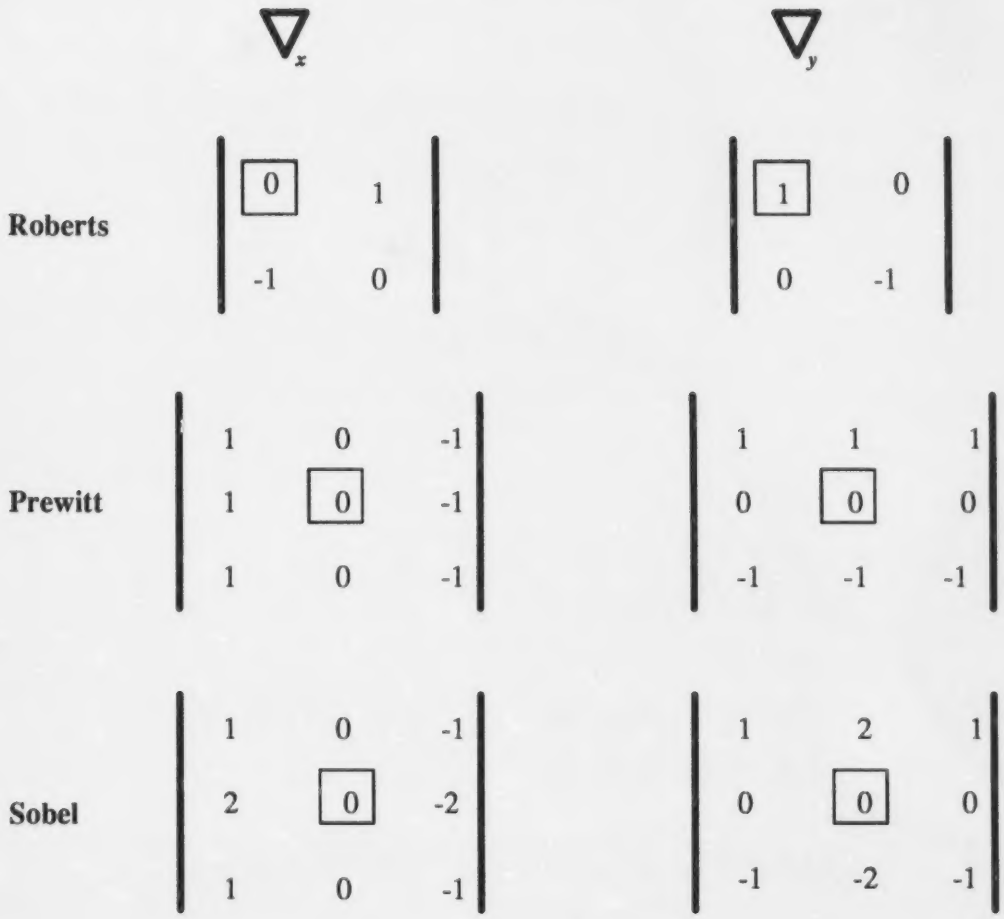


Figure 4: Common Gradient Operators

3.2.2 Compass Operators

Compass operators measure gradients in a selected number of directions. An example of compass operator is Kirsch which uses a 3×3 neighborhood to compute the gradient in eight possible directions. The mask used for calculating the gradient is given by

$$G(x,y) = \begin{bmatrix} 3 & -5 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & 3 \end{bmatrix}$$

Each clockwise circular shift of elements about the center rotates the gradient direction by 45°. The gradient $\nabla f(x,y)$ is thus given by

$$\nabla f(x,y) = \max(|G_i(x,y)|)$$

Larger templates can also be used with Kirsch operator. The advantage of using larger templates is increased smoothing and reduced noise sensitivity. However, larger templates are computationally expensive and hence avoided. Also, the same smoothing effect can be obtained by using the Sobel operator with higher weights. The use of weights enhances the computation over the central pixel of the window giving a smoothing effect over the rest. Another example of compass operator is the Frei and Chen operator [15] which is not discussed here.

3.2.3 Laplacian Operators and Zero Crossings

The Laplacian operator for a continuous function $f(x,y)$ is defined as

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

and can be approximated in the same way as the gradient operators for the discrete case. For 4-connected neighbors, the Laplacian is approximated by

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1+y) + f(x,y+1) + f(x,y-1) - 4 * f(x,y)$$

and for 8-connected neighbors it is given by

$$\begin{aligned}\nabla^2 f(x,y) = {} & f(x-1,y-1) + f(x-1,y+1) + f(x+1,y+1) + f(x,y+1) \\ & + f(x,y-1) + f(x+1,y-1) + f(x-1,y) + f(x+1,y) - 8 * f(x,y)\end{aligned}$$

The masks used for calculating the Laplacian are given by

$$\nabla^2 f(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \nabla^2 f(x,y) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The Laplacian operator has two disadvantages as an edge measure: (i) useful direction information is not available, and (ii) the Laplacian, being an approximation to the second derivative, doubly enhances any noise in the image [3]. A better utilization of Laplacian is to use its zero-crossings to detect the edge locations [22].

3.2.4 Marr and Hildreth Operator

Marr and Hildreth [32] suggest an edge operator based on the zero-crossings of a generalized Laplacian operator. The generalized Laplacian operator is given by sampling the kernel

$$A\left\{1 - k\frac{x^2+y^2}{\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)\right\}$$

at each image point (x, y). The parameter σ controls the width of the Gaussian kernel and value of k is adjusted such that the sum of the elements of a given mask size is zero. Zero-crossings of a given image, after convolving it with the kernel, give its edge locations.

In general, the Marr and Hildreth [32, 31] edge detection scheme is based on a filtering step consisting of a 2-D symmetric Gaussian, followed by the localization of zero-crossings of the Laplacian of the filtered image. This operator performs rather well, but its optimality was not rigorously proved [47]. Also, the operator has the advantage that it automatically forms closed boundaries.

3.3 Thresholding

After the edges are extracted, thresholding is performed to select the meaningful edge elements. Thresholding is thus useful to filter out the weak edges, the edges having a low edge strength. This operation removes some of the

spurious edges arising due to noise.

The two most widely used approaches for thresholding an edge image are: fixed cut-off technique and adaptive thresholding. When a fixed threshold is used, all edge pixels characterized by edge strength less than the cut-off value are discarded. If $f(x, y)$ is the given picture having a gray-level range $[z_1, z_k]$ and T is any number between z_1 and z_k, the result of thresholding $f(x, y)$ at T is the binary picture defined by [16, 41]

$$f_t(x, y) = \begin{cases} 1 & \text{if } f(x, y) \geq T \\ 0 & \text{otherwise} \end{cases}$$

However, using fixed thresholding is not a very good method. Such a technique is noise sensitive and its result depends on the quality of the input image. For example, this approach may delete important information in relatively clean images and retain spurious edges in noisy images.

An adaptive threshold on the other hand is relatively independent of the noise present in the image. In this technique, a certain percentage of the edge pixels are retained after thresholding.

Another method utilizes the average edge strength of a contour over part of its length to remove the redundant information from the edge image. This technique was used successfully by Pentland with Marr-Hildreth zero-crossings [6]. If the average is above the threshold, the entire segment is retained. If the average is below cut-off level, no part of the contour appears is the thresholded image. This procedure is quite useful to eliminate weak

contours which may arise due to noise.

Chapter 4

Boundary detection

4.1 Limitations of previous approaches

There are two major drawbacks in most previous algorithms for boundary detection. First, they are inherently sequential in nature. Processing starts at a given pixel (or a set of pixels) and proceeds one pixel at a time according to a specified set of rules. This implies that similar operations cannot be performed simultaneously all across the image. This is undesirable for most real time applications which require a high recognition rate. The VR approach on the other hand is inherently parallel. The decision made at any pixel does not depend on the actions taken at the neighboring pixels. Thus all image pixels can be processed in parallel without affecting the output.

The second limitation of several boundary following methods lies in their

inability to take a "rough look" at a large neighborhood of a pixel, while processing using a small mask. This shortcoming can result in noisy contours being detected, as well as meaningful contours being dropped. Also, in the conventional boundary following algorithms tracking proceeds along the best path at each step and once a point is selected to be on boundary, the decision is never reversed. Due to the small size of the mask, a traditional algorithm may choose a wrong path resulting in spurious boundary. Although feedback mechanisms can be used [30] to overcome this problem, these are not very effective if the image is noisy. On the other hand, if information about the large neighborhood surrounding the pixel is available, the task of selecting the true boundary points becomes easier and the boundaries so obtained are largely independent of the noise present in the image. The VR algorithm is designed to avoid the drawbacks of the existing algorithms. Noisy contours are largely ignored by the proposed method, which can also fill in small missing gaps in significant boundaries.

4.2 Variable-resolution masks

The variable resolution approaches to improve current thinning and boundary detection algorithms were designed. The basic concept behind the proposed method is the variable resolution mask. These masks (templates) are partitioned into 9 equal square-shaped parts as shown in Figure 5. The cen-

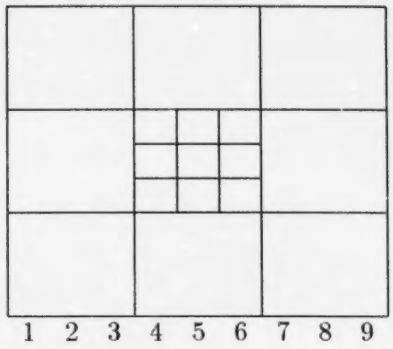


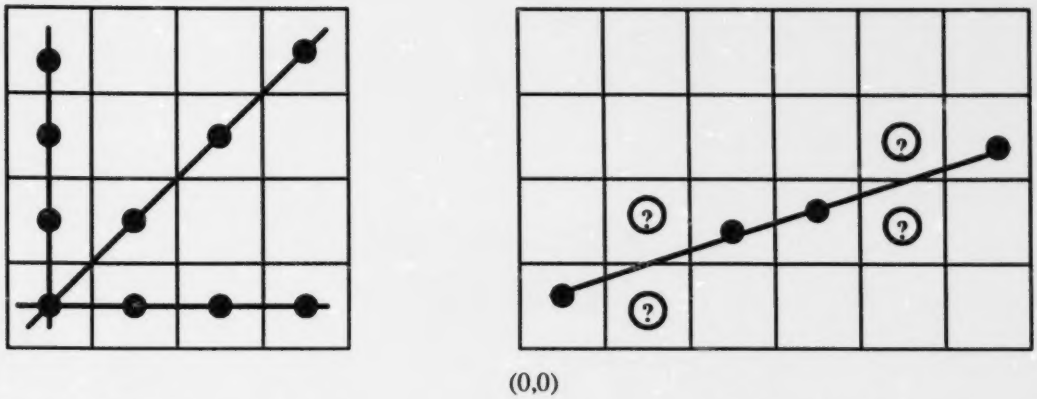
Figure 5: A variable resolution template

ter part is a regular 3×3 template. Each one of the eight peripheral parts also covers a 3×3 region, but at a reduced resolution. Thus the resulting 9×9 window is called a variable resolution (VR) window.

The VR masks are used as follows: First the result of the application of the high resolution center is examined. Then, if certain conditions are satisfied, some of the low resolution peripheral cells are used for further testing. Variable resolution templates have two advantages. First, the peripheral cells can be used to obtain a "rough" look at a large neighborhood around a given pixel. Second, the computational cost involved in the process is only marginally higher than the cost of using small uniform resolution templates. It is also important to note that the techniques described here are inherently parallel. That is, the decision taken based on the computations performed in one peripheral cell is completely independent of the decisions made in any of the adjacent regions.

4.2.1 VR Masks for Boundary Detection

The aim is to detect the contours in all the orientations. If a 3×3 mask is used then the maximum number of directions that can be checked for locating the next boundary point is 8, assuming the pixel (i, j) is 8-connected. Since the VR technique uses a larger mask of 9×9, it can be used to look for the lines in 16 directions. The extensive experiments conducted on various images prove that use of 16 directions is enough to detect the contours of all orientations. To design the variable resolution masks for the lines of various orientations we use a simple technique from computer graphics. Since straight lines should appear as straight lines with constant brightness, along their length, the selection of the pixels along the length of the line should be given special consideration. Horizontal, vertical and 45 degree lines are easy to detect as there is no conflict in selection of the pixel to be highlighted. However for lines of all other orientations, an optimum raster location should be selected since not all lines pass precisely through a raster point. This is illustrated in the Figure 6. The pixels which lie on the line are shown by "•" in the figure. Figure 6(a) shows the rasterization of horizontal, vertical, and diagonal lines. As it can be seen, there is no conflict in proper selection of the pixels that lie on the lines. Figure 6(b) shows a line of 22.5° orientation which does not pass exactly through a raster point. The line starts at $(0, 0)$ and subsequently crosses 6 pixels. To select the right pixel, let us assume that the pixels are represented by a square on the square grid. Since we have a VR mask of 9×9, our center pixel is at location $(0, 0)$. To find out the pixels lying on the line of 22.5° orientation, the basic idea from trigonometry is utilized. The



(a) Rasterization of horizontal, vertical and 45 degree orientation lines

(b) Rasterization of a line of 22.5 degree orientation

Figure 6: Rasterization of Straight Lines

end point of the line will be at location $(4, y)$. Now the y intercept of the line is given y the following equation

$$y = \lceil x \tan\theta \rceil$$

where θ is the orientation of the line. To find out the pixels lying on a line of particular orientation, θ is kept fixed, while the x-intercept keeps on changing. For $\theta = 22.5$ and $x = 4$ a value of y=2 is obtained implying that the end point should be at location $(4, 2)$. Similarly, other pixels lying on the line whose end points are at $(0, 0)$ and $(4, 2)$ can be found. For this, either a simple line rendering technique from computer graphics can be used or the above equation can be utilized. In this case, the above equation was used to derive the other points which will lie on the line. The points so obtained are shown in the Figure 7. This figure corresponds to the VR mask for detecting lines of 22.5° orientation. The symbol "sf x" in the figure denotes

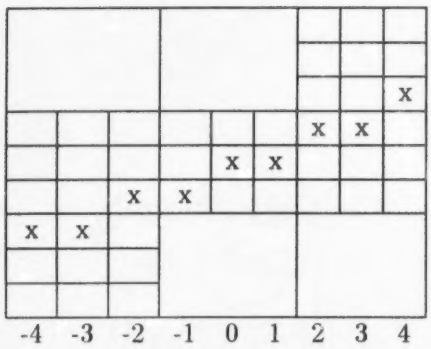


Figure 7: Mask for 22.5° orientation line

the pixels which line on a line of 22.5^{circ} orientation. Similarly the masks for all other orientations can be designed. The masks for detecting lines in all the orientations are given in the appendix A.

4.3 A VR approach to boundary following

In this section the VR boundary detection algorithm is presented. The method essentially consists of two steps: the edge detection and thresholding step followed by the boundary construction step.

The main purpose of the first step is to highlight those points which lie in the neighborhood of the boundaries. Let $I(x, y)$ be the given image. The edges are detected by convolving the image with the second derivative of the Gaussian and extracting the zero crossings. Two-dimensional Gaussian with

a standard deviation σ is defined by the function

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Thus it is needed to find the zero-crossings in

$$f(x,y) = \nabla^2 [G(x,y) * I(x,y)]$$

where "*" is the convolution operator. However, using this method does not give the gradient strength and orientation of the zero-crossings. To obtain the gradient strength and the gradient direction, the two-dimensional Gaussian is separated into the product of two one-dimensional functions given by

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

$$G(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

The first derivative of the Gaussian is computed for every point in the image and directional derivatives I_x and I_y are obtained. From these directional derivatives the strength and direction for each zero-crossing pixel is computed by the following

$$s(x,y) = \sqrt{{I_x}^2 + {I_y}^2}$$

$$\phi(x,y) = \tan^{-1}(I_y/I_x)$$

An adaptive thresholding technique is used to delete the zero crossings with very low gradient strength. The advantage of using an adaptive threshold is that it is relatively independent of the quality of image. The pixels which survive thresholding are assigned weights depending on the gradient magnitude and gradient direction. For example, a pixel is assigned the maximum weight if it has a high edge strength and difference between the gradient

angle with the adjacent pixel is minimum. In the second step, the pixels are linked together, to get the meaningful boundaries, using the VR boundary detection algorithm. The VR boundary detection algorithm can be described by the following pseudocode:

```
procedure VR_Boundary_Detection

for   (for all image neighborhoods) do
if   (center pixel of small window has maximum weight)
{              *** step 1 ***
  check 3 × 3 neighborhood for edge in vertical,
  horizontal and other directions;
  if (no edge is found in the small window)
  {              *** step 2 ***
    take a "rough" look at appropriate peripheral cells;
    if (larger window suggests the presence of an edge)
      Mark the appropriate pixels in the smaller window
      as edge pixels to get a continuous boundary;
  }
}
```

For the purpose of assigning weights, the edge image was thresholded to a three-valued image depending on the edge strength. We used an adaptive threshold to classify a certain percentage of pixels as *High*, *Medium*, and *Low*. Pixels in the percentile ranges [0,25], [25,60], [60,100] are classified as low, medium, and high, respectively. Extensive experimental results suggest the use of the above thresholds. Similarly, the gradient angle is digitized into 16 different directions. The total weight is defined as the sum of the following:

1. Strengths classified as *High*, *Medium*, and *Low* are assigned weights 3, 2, and 1, respectively.

2. A weight of 3 is assigned if the difference in gradient direction of the two adjacent pixels is less than 45 degrees. Similarly a weight of 2 is assigned if the direction difference is between 45 degrees and 90 degrees, a weight of 1 is assigned if it is between 90 degrees and 180 degrees, and a weight of zero is assigned for direction difference greater than 180 degrees.

Based on this scheme the maximum weight a pixel can have is 6 (it has a high edge strength and a low direction difference) and the minimum weight is 1 (it has a low strength and a high direction difference). The algorithm uses the high resolution central window to detect the boundary pixels in all the 16 possible directions. Only if it is unable to decide which boundary pixels are suitable, does it look at the outside low resolution periphery.

The information in the peripheral cells is reduced in a manner similar to VR thinning. By having a rough look at a larger neighborhood, we can fill in the small gaps and also avoid small noisy contours. The method is inherently parallel as each window is independent and can therefore be processed simultaneously.

Two steps in the above pseudocode need further explanation – how the information in the peripheral cells is reduced, and what is the criteria of making a decision in the high resolution central window? Peripheral cells are reduced to a single bit, α. This bit essentially answers questions such as: "is there a vertical edge?", "is there a horizontal edge?", "is there a diago-

p_4	p_3	p_2
p_5	p_0	p_1
p_6	p_7	p_8

(a) 3 x 3 template

p_{44}	p_{43}	p_{42}	p_{34}	p_{33}	p_{32}	p_{24}	p_{23}	p_{22}
p_{45}	p_{40}	p_{41}	p_{35}	p_{30}	p_{31}	p_{25}	p_{20}	p_{21}
p_{46}	p_{47}	p_{48}	p_{36}	p_{37}	p_{38}	p_{26}	p_{27}	p_{28}
p_{54}	p_{53}	p_{52}	p_4	p_3	p_2	p_{14}	p_{13}	p_{12}
p_{55}	p_{50}	p_{51}	p_5	p_0	p_1	p_{15}	p_{10}	p_{11}
p_{56}	p_{57}	p_{58}	p_6	p_7	p_8	p_{16}	p_{17}	p_{18}
p_{64}	p_{63}	p_{62}	p_{74}	p_{73}	p_{72}	p_{84}	p_{83}	p_{82}
p_{65}	p_{60}	p_{61}	p_{75}	p_{70}	p_{71}	p_{85}	p_{80}	p_{81}
p_{66}	p_{67}	p_{68}	p_{76}	p_{77}	p_{78}	p_{86}	p_{87}	p_{88}

(b) a 9 x 9 template

Figure 8: (a) a 3×3 template (b) a 9×9 template

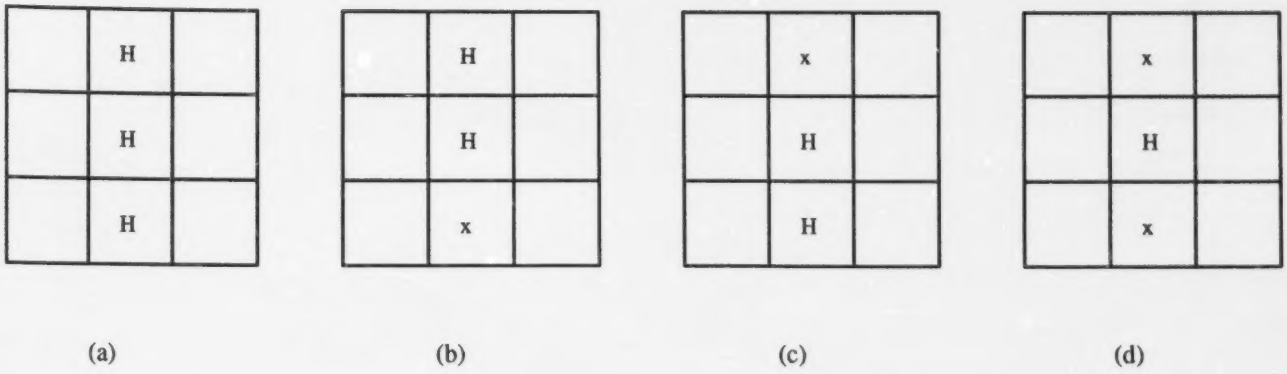


Figure 9: Example Templates for detecting a vertical edge

nal edge?", or is there an edge of any other orientation, for example, 22.5° angle. This term α is obtained by using a simple technique for the relevant peripheral cell. For example, for locating a vertical edge, a thresholding is used to determine α as

$$\alpha = \begin{cases} 0 & \Sigma(p_{30} + p_{33} + p_{37}) \leq 15 \\ 1 & \text{otherwise} \end{cases}$$

Similarly, α can be determined for locating edges of other orientations by selecting the relevant pixels in the peripheral cells. Figure 8(b) shows a 9×9 neighborhood and the technique used to identify each pixel.

It is worth mentioning at this point that this larger 9×9 template is used only if no decision about the center pixel, p_0 of the 3×3 high resolution window, being on the boundary is made on step 1 of the algorithm. Figure 8(a) shows the 3×3 template. To explain the decision criteria used

in step 1 of the VR technique, consider Figure 9(a). The three pixels "H" pixels forming a straight line in the 3×3 region shown in the figure indicates the presence of a vertical edge. In such a situation, the larger 9×9 template is not checked and the pixels marked with "H" in the figure are declared as the boundary pixels. Now consider the Figures 9(b-d). The pixel marked with "x" indicates a *don't care condition*. This necessitates a check on the relevant peripheral cells to detect the presence of a vertical edge. A value of $\alpha = 1$ indicates a vertical stroke. Similarly, the edges of all other orientations can be detected.

4.3.1 Complexity of the Algorithm

The VR boundary detection algorithm involves two steps:

1. Checking the high resolution central window to check if the pixel forms a part of the boundary.

2. Looking at the peripheral cells in case no decision about the center pixel of high resolution window being on the boundary is made based on step 1.

The center pixel is marked to be on the boundary if either (1) holds or (2) holds. It is worth mentioning here that if (1) is true than (2) is not executed. Thus the worst case complexity of the VR algorithm equals the complexity of step (1) plus the complexity of step (2).

Let I be the image, an $N \times N$ array of pixels each taking one of 256 gray values; M be the template, which is $n \times n$ with $n << N$. Each of the $n \times n$

subimages of I can be uniquely referenced by its upper left corner coordinates (i, j). Thus there are $(N - n + 1)(N - n + 1)$ such (i, j)s. If a template of size $n \times n$ is used, $n^2 - 1$ pixels are involved. Since we are checking for 16 different directions in step (1), the complexity would be $8(n^2 - 1)$. Further if step (2) is to be executed, we at the most look at a total of $2n$ pixels in the peripheral cells, which implies that the complexity of step (2) is $8(2n)$. Hence the worst case complexity of the algorithm would be

$$\left\{\Big(8(n^2 - 1) + 8(2n)\Big)\ (N - n + 1)^2\right\}$$

Since n is much smaller than N, the effective complexity of the algorithm is $(8(n^2 + 2n - 1)N^2)$ which is $O(n^2N^2)$. Thus the use of the VR mask does not increase the order of the complexity which remains the same as when a uniform resolution template of $n \times n$ is used. However, the use of VR templates increase the total number of computations required. From the above analysis we see that the VR boundary detection algorithm will need at most 1.6 times the number of computations required by methods using 3×3 windows. It is important to note that the above analysis is for the sequential implementation of the algorithm.

4.4 Experimental results

Two-dimensional images were generated for testing the boundary detection algorithm. Test images were prepared using additive Gaussian noise at varying levels. In all, five levels of noise are represented in the test images. Gaussian noise is additive, with a mean of 0 and a standard deviation, denoted by η, of 4, 8, 16, 32, and 64. The test images contained 256 gray levels and had objects of both regular and irregular shapes. In addition to the synthetic test images, the algorithm was tested on several real images to verify its robustness. All the images were 256×256 pixels large.

Both test images and real images were subjected to two versions of the boundary detection algorithm. The first version utilized only the mean gradient magnitudes in the computation of costs while the second version combined mean gradient magnitude and gradient direction. Both versions of the algorithm were tested with various values of σ and different thresholds. Finally, an optimal value of $\sigma = 2$ was chosen based on the experimental results. Figure 10(a) shows the image of a cup. The figure has several well-defined edges. Figure 10(b) shows the edges extracted from Figure 10(a) and 10(c) gives the remaining edges after thresholding. Figure 10(d) shows the result of the VR algorithm.

In all, the algorithm was tested on nearly 100 different images. Typical results obtained are shown in the figures. It was observed that version 2 of the algorithm, incorporating both the gradient direction and the gradient magnitude, gave slightly better results on images with very high levels of noise ($\eta \geq 32$). For comparison with the proposed method, two previous

Figure 10: From the top left corner in clockwise direction: Original Image of a cup, Edge Image, Image after Thresholding, Result of VR algorithm

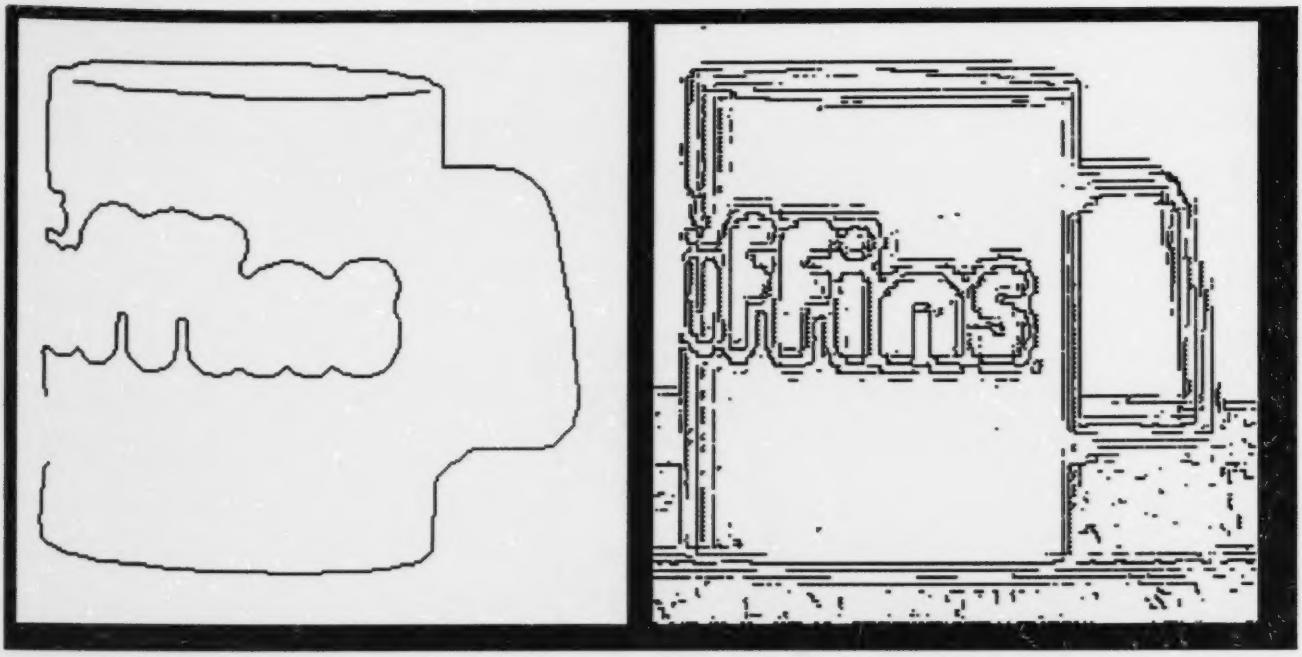


Figure 11: Results of Williams' method (left side) and Lacroix's method (right side)

algorithms [51, 27] were chosen. The first module of the algorithm given in [27] was implemented using the local operator with unweighted masks, as described in the paper. In [27], a test on the length of the contour was performed to delete edges less than 4 pixels in length. We implemented the algorithm given in [27] without performing the aforementioned test in order to maintain compatibility with our algorithm which retains short contours. The source code for the other algorithm [51] was obtained from its authors. The same values of σ mentioned in the paper [51] were used. However, a different threshold of 0.25 had to be applied as the algorithm failed to give good results with a threshold of 0.08 (as stated in [51]).

The cup image as shown in Figure 10(a) was also subjected to Williams' and Lacroix's methods. The results obtained are shown in Figure 11. Lacroix's method failed completely on this image as several double contours were obtained. This happened because the number of pixels retained after thresholding was quite large and thus resulted in redundant contours — one drawback of the Lacroix method which was observed in all the images. The quality of the boundaries depended directly on the threshold. Though the paper claimed the use of an adaptive threshold where only a certain percentage of the pixels, say x percent, were retained after the edge detection phase, the value of x depended entirely on the quality of image. Most of the images gave good results when only 10-15 percent of the pixels were retained. However in some images it resulted in loss of information while in others, for example in the cup image, it resulted in double contours. Williams' method gave good results on this image though it missed out some of the details. Figures 12– 16 present the results on artificial test images generated. Figures 17– 21 show the results on the real images.

Figure 12 shows the artificial image of a gray-level band. For a clean image with no noise, all the algorithms were able to extract perfect boundaries. However as the noise level increased in the image, the boundaries obtained by the Lacroix and Williams methods did not give straight lines. On the other hand, the VR method was still able to obtain straight lines for even high noise levels ($\eta \geq 16$). For $\eta \geq 4$, Williams' method was unable to obtain a closed boundary. For noise levels greater than $\eta = 16$, both Williams' and Lacroix's methods gave poor results whereas the VR algorithm retained all the boundaries.

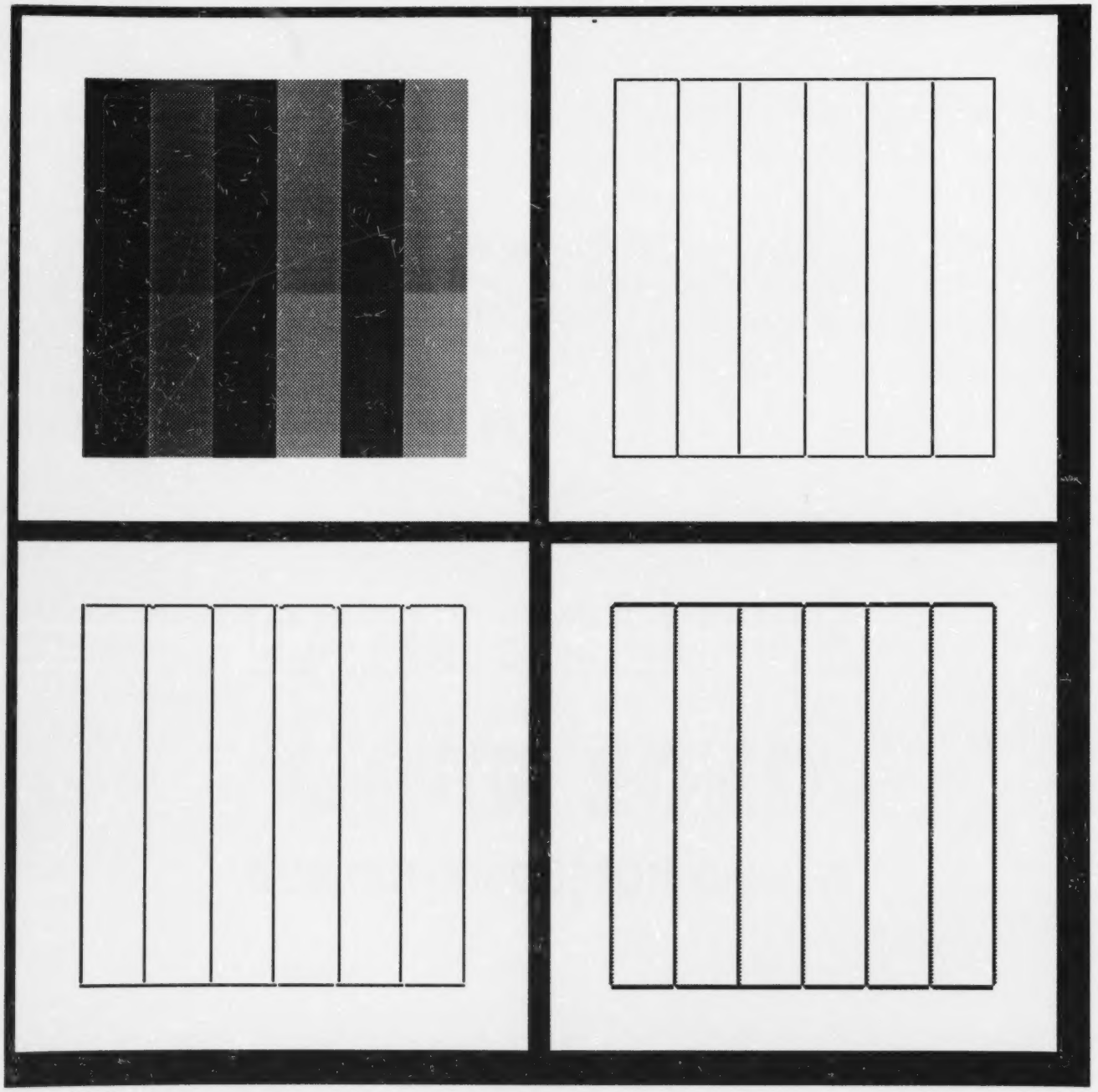


Figure 12: Test Image with no noise. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method.

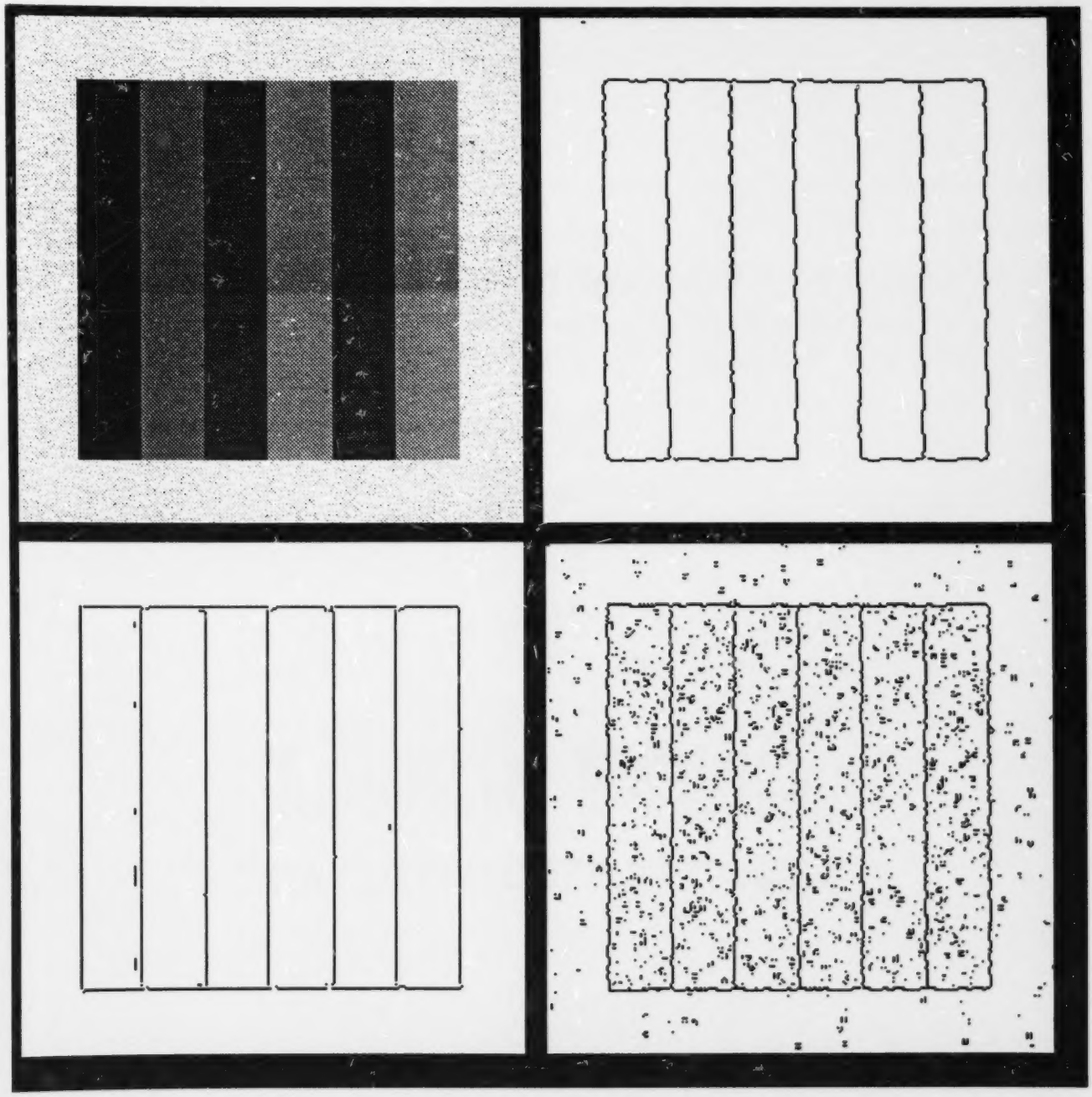


Figure 13: Test Image with Gaussian Noise of $\mu = 0$, $\eta = 8$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method.

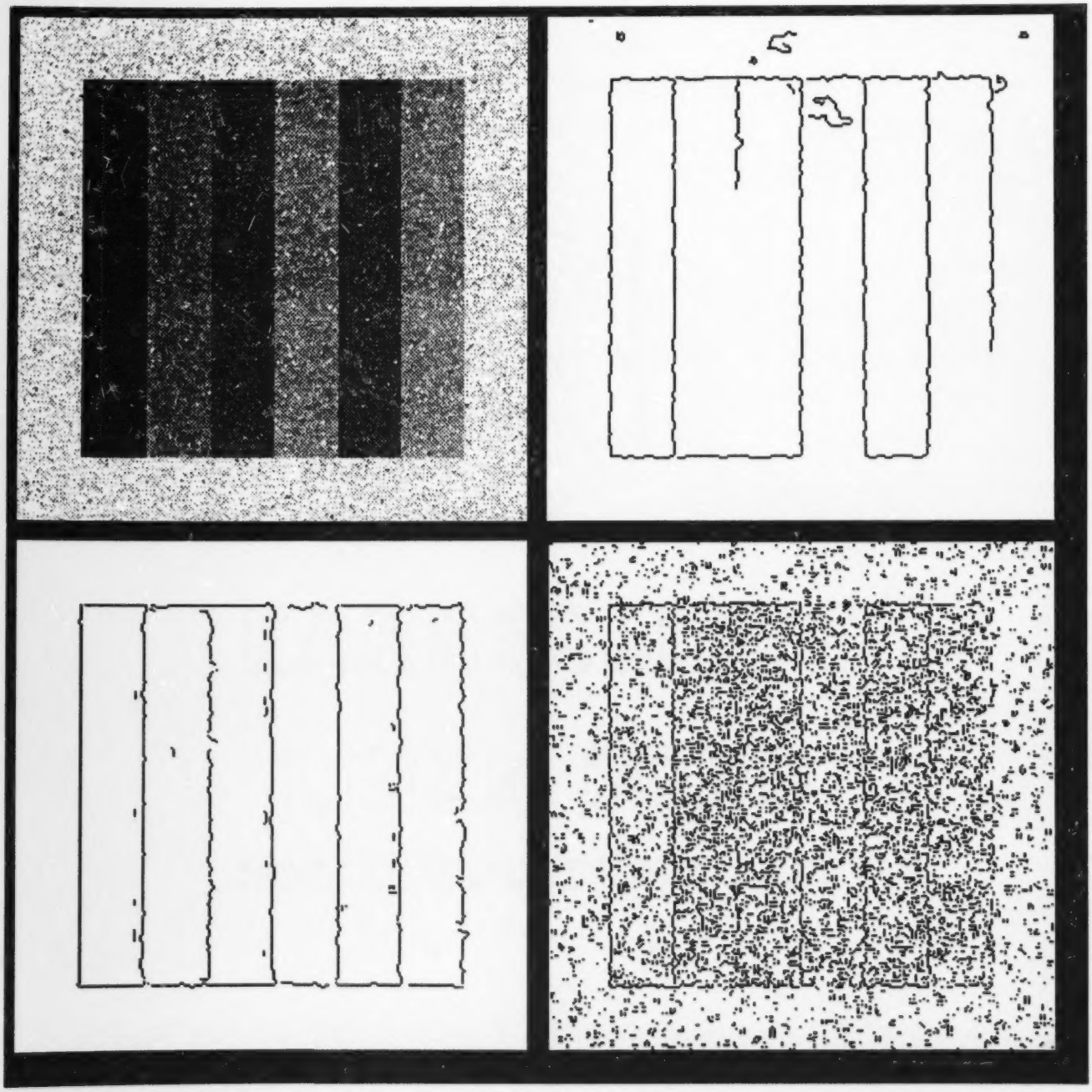


Figure 14: Test Image with Gaussian Noise of $\mu = 0$, $\eta = 32$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method.

Figure 15 shows an artificial image of a cone intersecting a triangle. For this image Williams' method failed to obtain all the contours, especially the contours of the cone. This phenomenon was also observed on some other images where Williams' method failed to extract all the boundaries. Lacroix's algorithm performed quite well on images with a low level of noise but its performance deteriorated as the noise level increased. The method failed to obtain meaningful boundaries for the noise level of $\eta = 32$. On the other hand, the VR algorithm gave consistently good results. Even for the noise level of $\eta = 32$, the VR method was able to obtain fairly clean boundaries. Figures 15– 16 show the results of the various algorithms.

Now some experiments with real images will be described. First results with a different cup image – one with sharply defined edges – are shown. Williams' method obtained good results on this image but missed out one of the boundaries. Lacroix's method, on the other hand, was able to obtain the whole cup boundary but missed out other relevant edges. The VR algorithm was able to pick up the complete boundary details as well as the other details on the surface of the cup. Figure 17 shows the results of the three methods.

Similarly, Figure 18 shows the results of the three methods on the image of a book. Here both Williams' and VR methods obtained better results than Lacroix's method. For very noisy images, the VR method obtained better results than the other two methods. Figures 19– 21 show the results of the three methods on images with a high degree of noise. Here the VR method was still able to obtain a sufficiently large number of contours unlike Williams' method. Lacroix's method generated several noisy contours in addition to the meaningful ones.

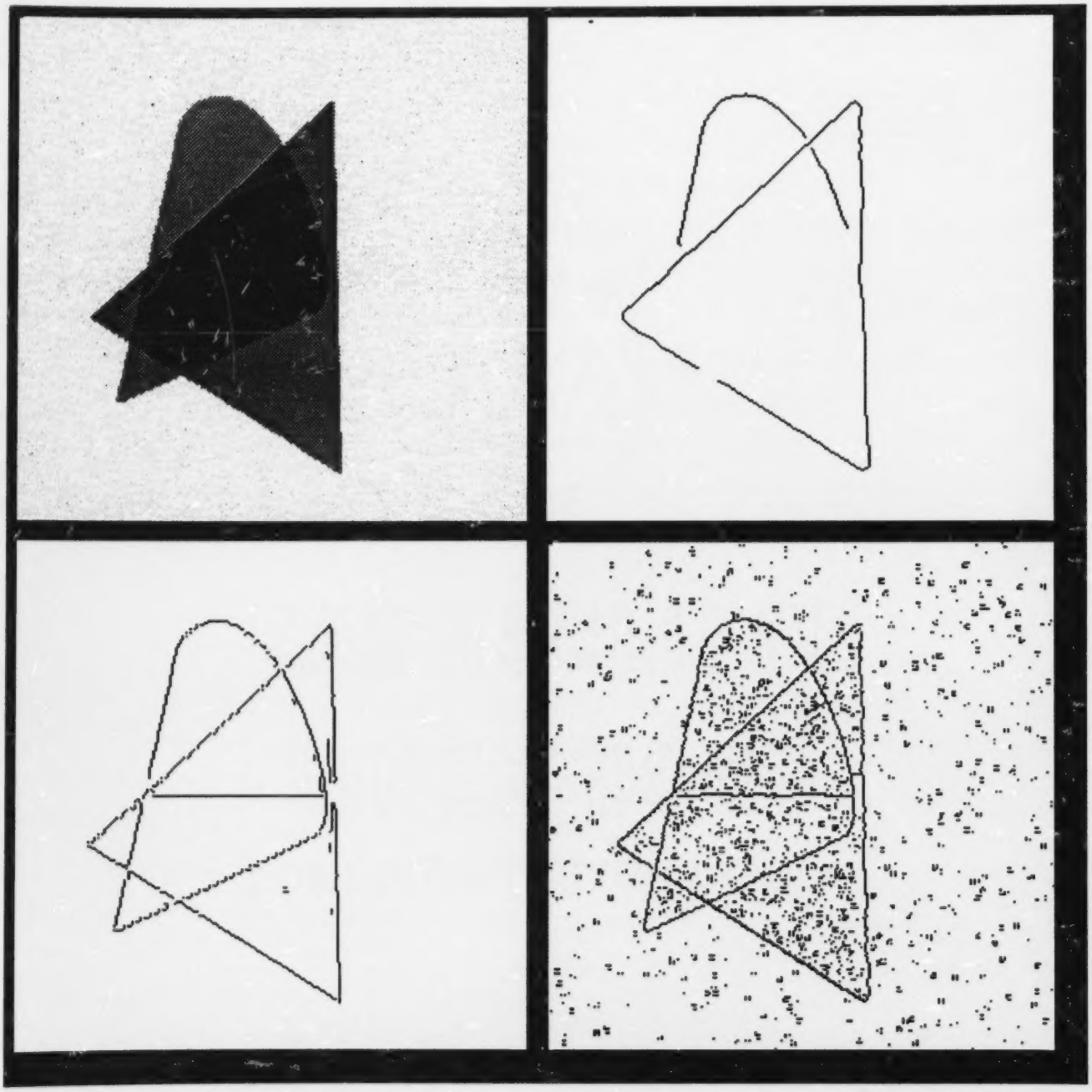


Figure 15: Test Image with Gaussian Noise of $\mu = 0$, $\eta = 4$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method.

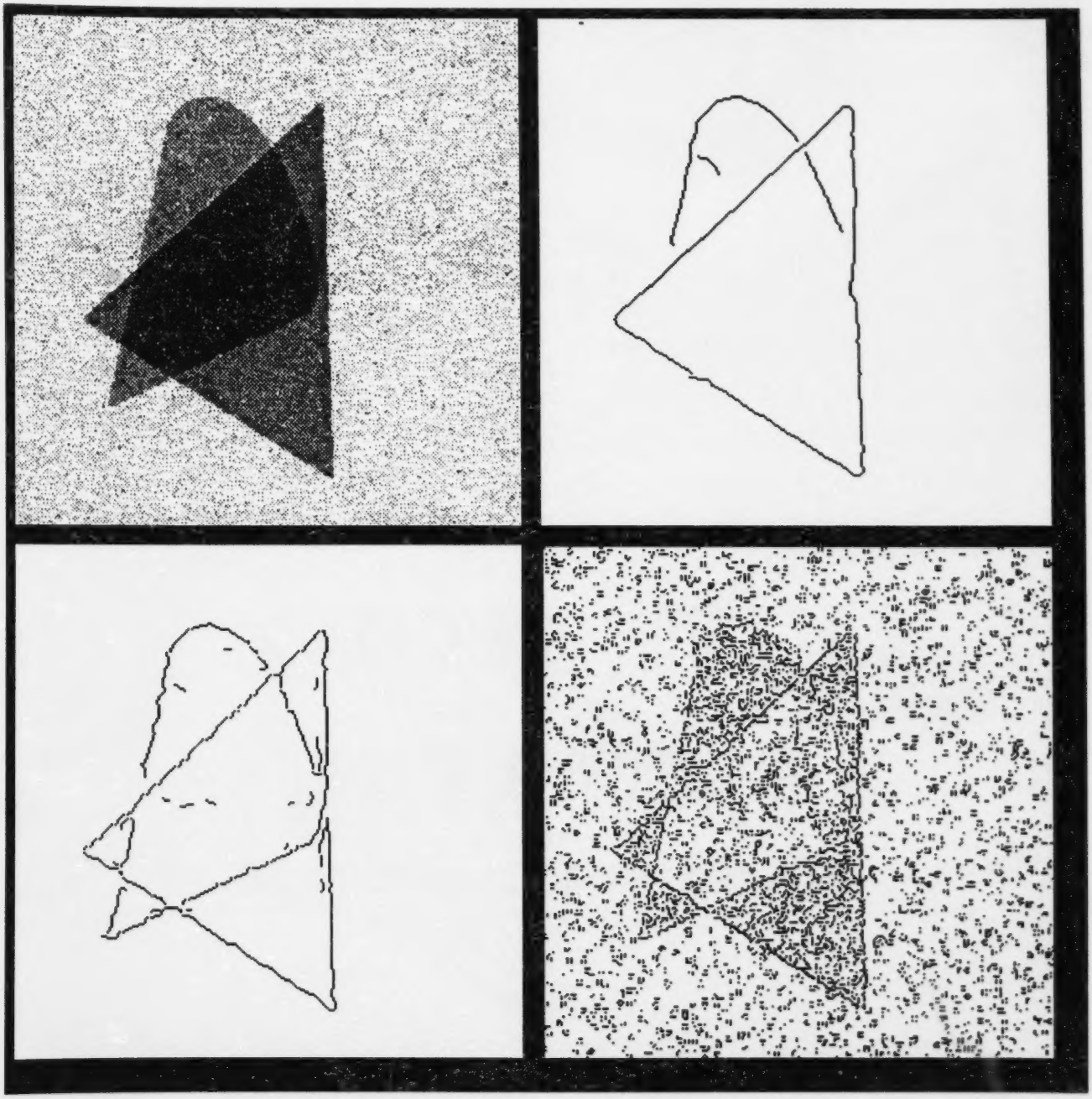


Figure 16: Test Image with Gaussian Noise of $\mu = 0$, $\eta = 32$. From the top left corner in clockwise direction: Original image, Williams' method, Lacroix's method, VR method.

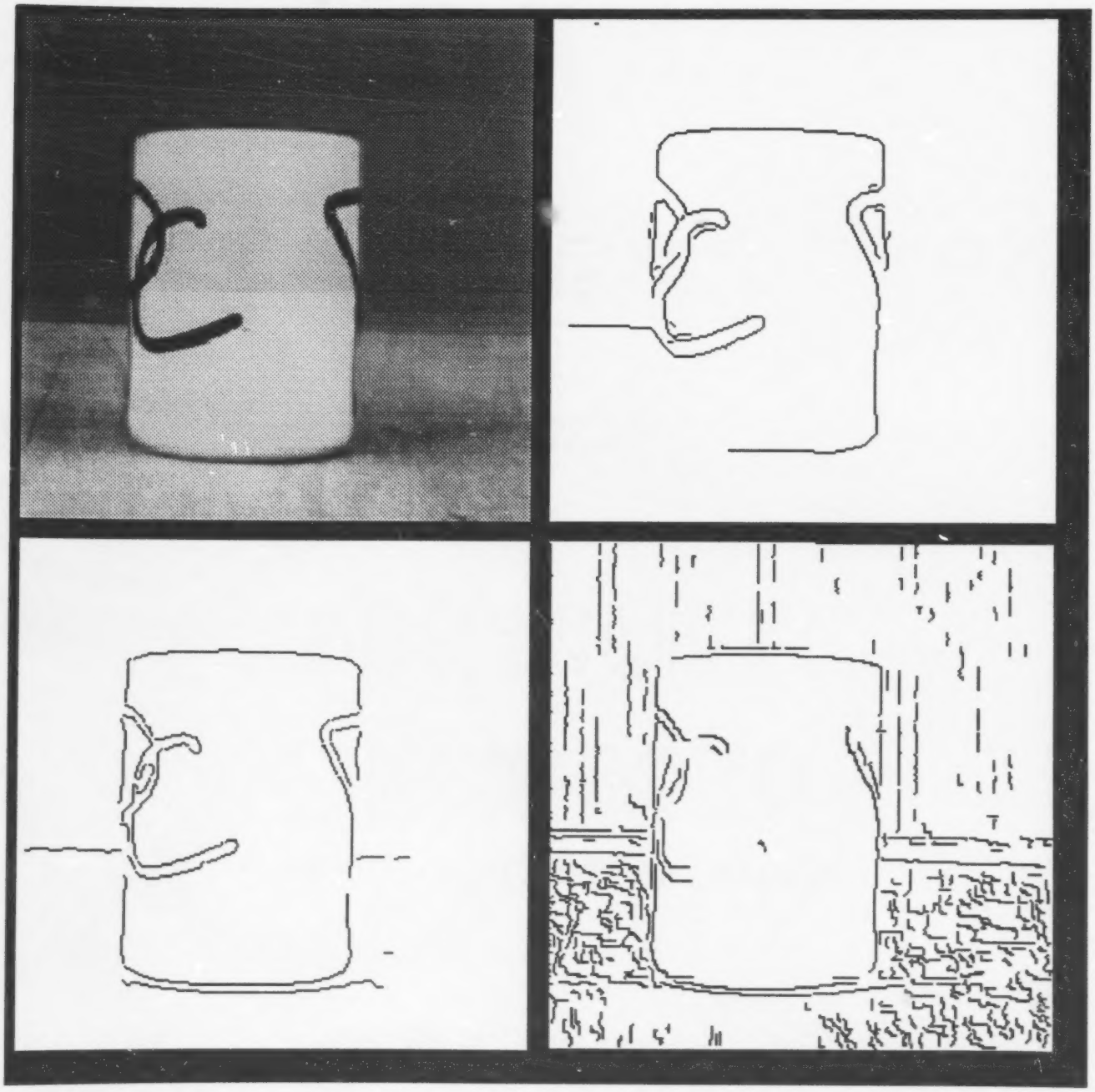


Figure 17: From the top left corner in clockwise direction: Original Image of the cup , Williams' method, Lacroix's method, VR method

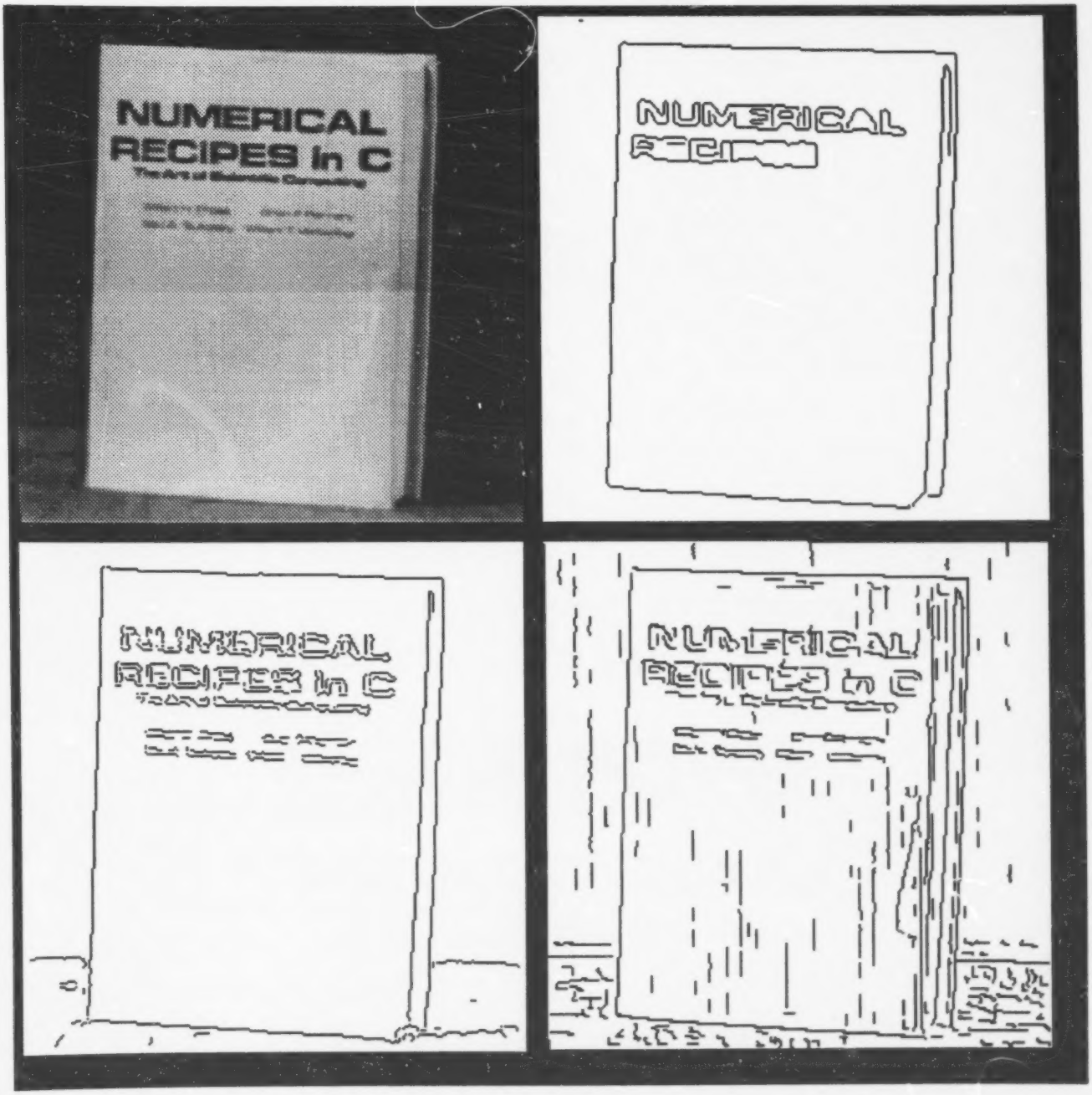


Figure 18: From the top left corner in clockwise direction: Original Image of the book, Williams' method, Lacroix's method, VR method

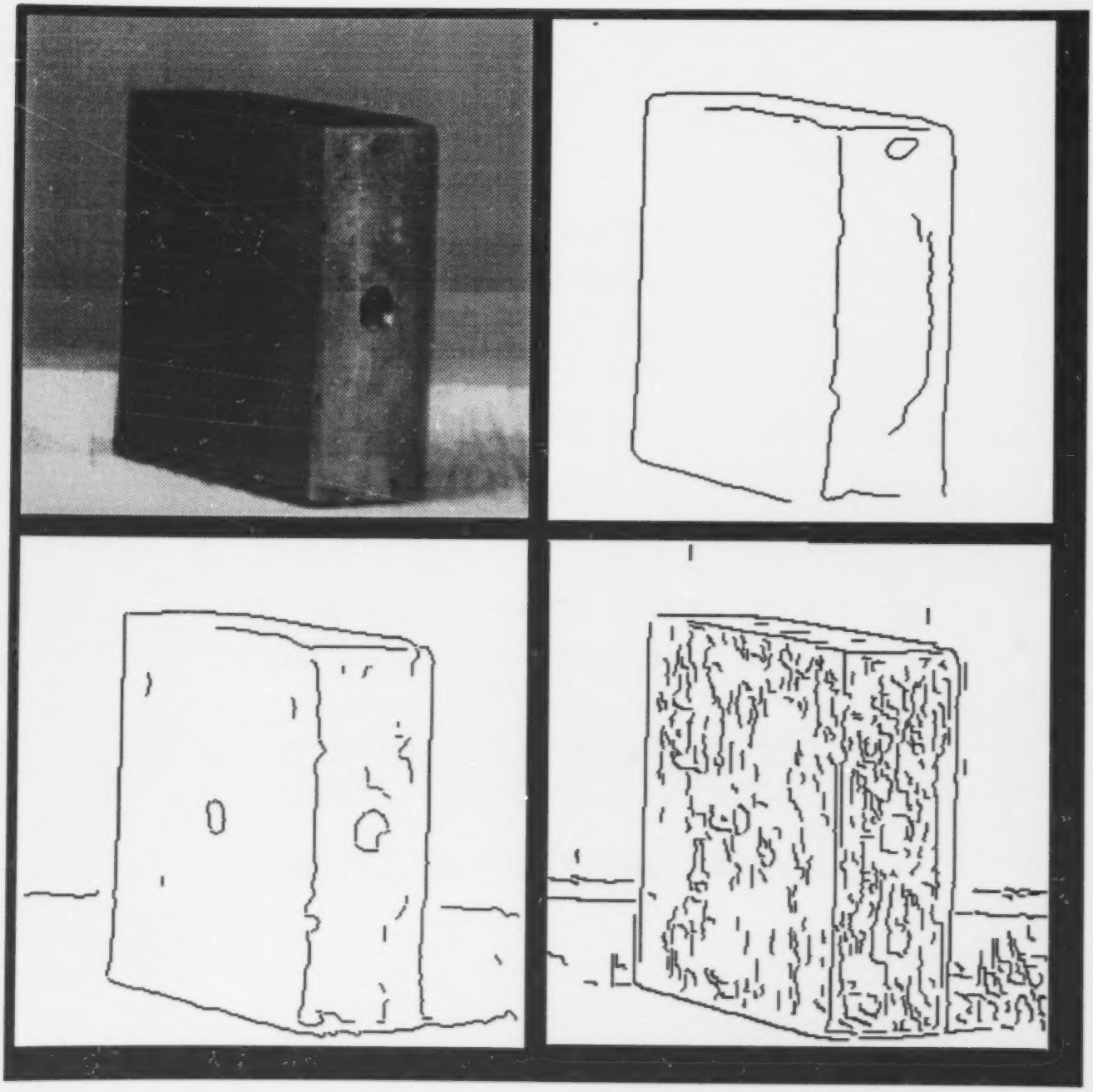


Figure 19: From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method

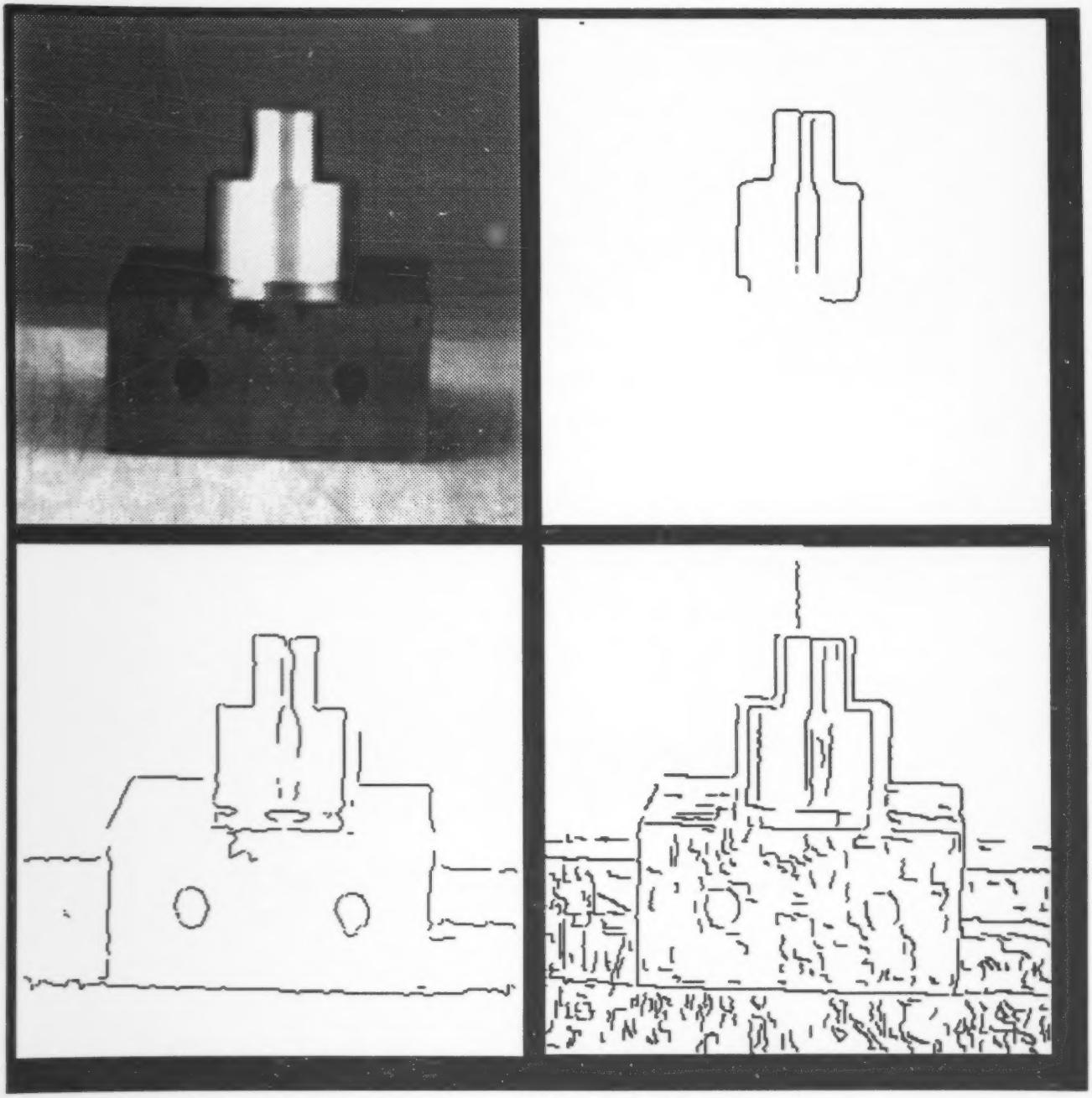


Figure 20: From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method

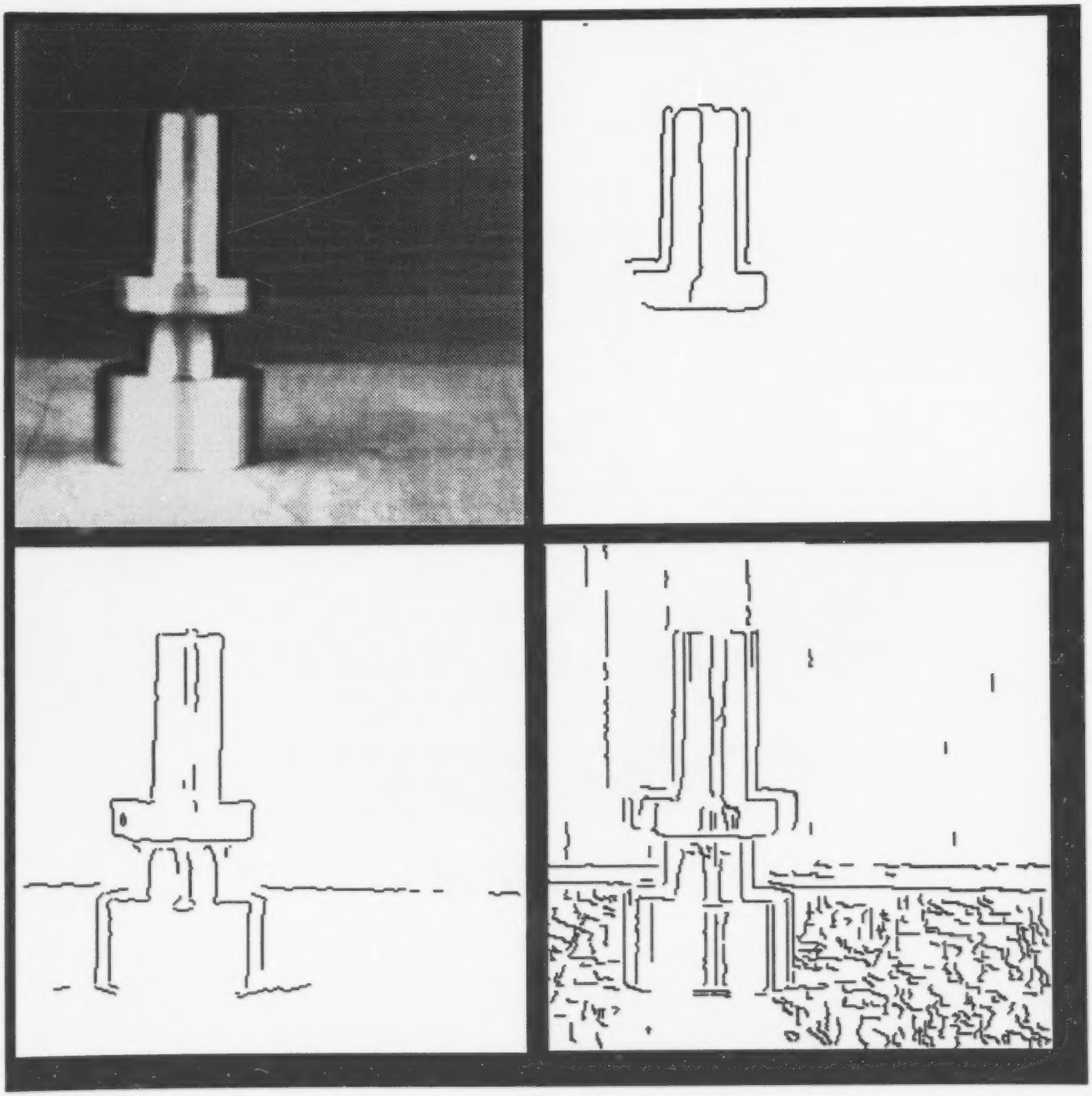


Figure 21: From the top left corner in clockwise direction: Original Image of a block, Williams' method, Lacroix's method, VR method

Chapter 5

Character thinning

5.1 Limitations of small templates

The following examples demonstrate the severe limitations of current thinning algorithms using small templates. To illustrate this point consider the distortions caused on the character "B" by various well-known methods. Figure 22(a) shows the input character. A symbol "#" indicates a character pixel, i.e., a pixel with value 1. Background pixels with value 0 are shown as blanks for the ease of visualization. A human thinning operator would generate a skeleton with a straight vertical stroke at the left end. None of the computer thinning algorithms (using 3×3 templates) can achieve this.

The 3×3 neighborhood of pixel "Z" is given in Figure 22(b). Figure 22(c) shows the result of Guo's thinning algorithm A1 [19] after iteration 5. Consider the contour pixel marked "V" at the middle of the left-most column. Figure 22(d) gives the neighborhood of this pixel. Since only 3×3 templates

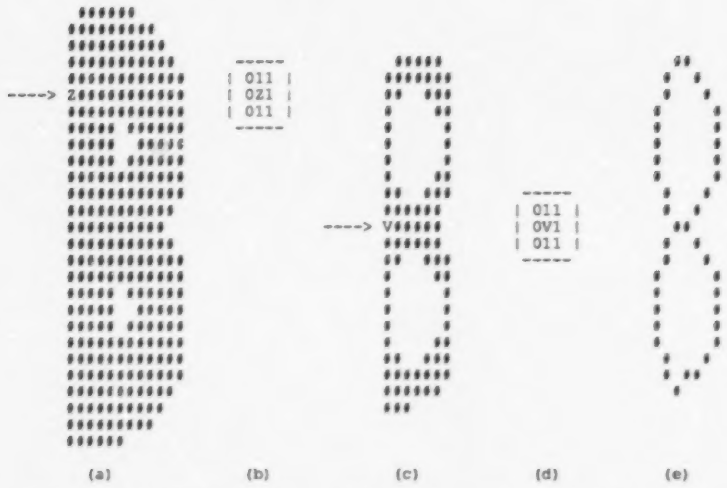


Figure 22: Character "B" and some thinning results using Guo's algorithm A1 (a) input, (b) neighborhood of pixel "Z", (c) after iteration 5, (d) neighborhood of pixel "V", (e) the final skeleton

are used, the matching result for pixel "V" is no different from that of pixel "Z". Thus pixel "V" will be deleted from the character in iteration 6, just as pixel "Z" was deleted in iteration 1. Several other pixels near pixel "V" (see Figure 22(c)) are in a similar situation to "V", and are also deleted. As a result, a "valley" is formed at the middle-left part of the character (Figure 22(e)). A character classification procedure is likely to recognize Figure 22(e) as the number "8" rather than the letter "B". Several existing thinning algorithms using 3×3 templates will generate a skeleton similar to this. This problem cannot be solved as long as only 3×3 templates are used.

Holt et al. [20] use a 4×4 window, which is still too small to correct the above problem. Figure 23 shows a similar phenomenon when Holt's method

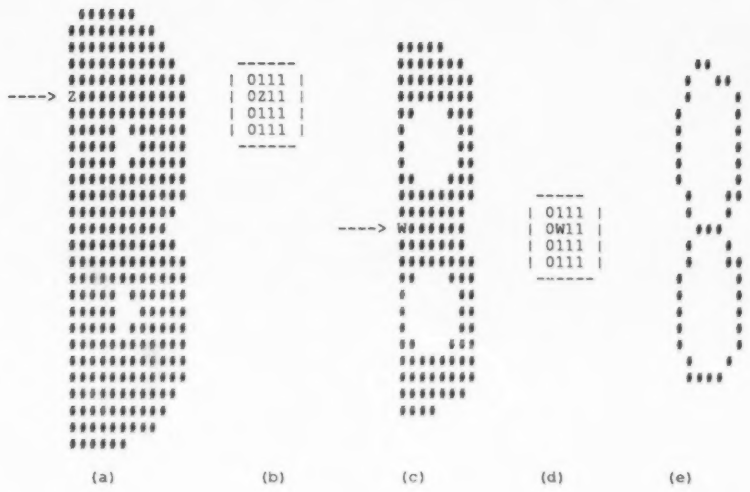


Figure 23: Character "B" and some thinning results using Holt's algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 2, (d) neighborhood of pixel "W", (e) the final skeleton

is applied to the same input character.

Figure 23(b) shows the neighborhood of pixel (10,5) which is marked "Z" in Figure 23(a). Figure 23(c) gives the result of iteration 2 of Holt's method. Pixel (18,8) is marked "W". Figure 23(d) gives the neighborhood of pixel "W", which is identical to that of pixel "Z". Figure 23(e) gives the final result.

The above examples demonstrate the fact that vertical strokes are critical in distinguishing between characters, such as "B" and "8", "K" and "X", "E" and "Σ". In processing common characters, a shape-preserving thinning algorithm should pay special attention to retaining vertical, horizontal, and diagonal strokes. This problem was first noticed by Govindan and Shiv-

aprasad [17] who proposed a sequential method to make thinning algorithms more shape-adaptive. In their method, contour-tracing algorithms are used to identify the presence of right-angles, T-corners and acute-angle corners. The pixels of the character contours are examined sequentially following a contour-tracing algorithm. The chain code of several previous pixels is stored to help in the decision about the current contour pixel. In the test for different corners a neighborhood larger than 3×3 is required. Wigena and Li [50] use part of a large template (5×5) to make thinning more shape-adaptive. It is becoming more obvious that a larger window is necessary to improve the quality of thinning, thus it is desirable to efficiently obtain information from a larger neighborhood of the contour pixels.

Shape adaptive methods use larger templates which either require larger memory or more complex testing. Because of the time and space limitation of current computer technology, 9×9 templates have not been used. This thesis proposes a simple *parallel* thinning algorithm which *efficiently* uses information from a large neighborhood to preserve the shape of the resulting skeleton.

5.2 Shape preservation using VR scheme

The variable resolution thinning algorithm can be best described by the following pseudocode:

```
procedure VR_Thinning

  repeat   (for desired number of iterations)
  {
   for   (for all image neighborhoods)   do
     if   (small window suggests deletion of pixel)
     {
       take "rough" look at appropriate peripheral cells;
       delete pixel only if there is no shape distortion;
     }
  }
```

Two steps in the above pseudocode need further explanation — how the information in the peripheral cells is reduced, and how the shape distortion is prevented using this information? Peripheral cells are reduced to a single bit, γ. This bit essentially answers questions such as: "is there a vertical stroke?", "is there a horizontal stroke?", or "is there a diagonal stroke?". In order to understand how γ is obtained consider two values, α and β, for the relevant peripheral cells. In the current experiment, a simple thresholding is used to determine α as

$$\alpha = \begin{cases} 0 & \text{if there are } \geq 5 \text{ character pixels in this region} \\ 1 & \text{otherwise} \end{cases}$$

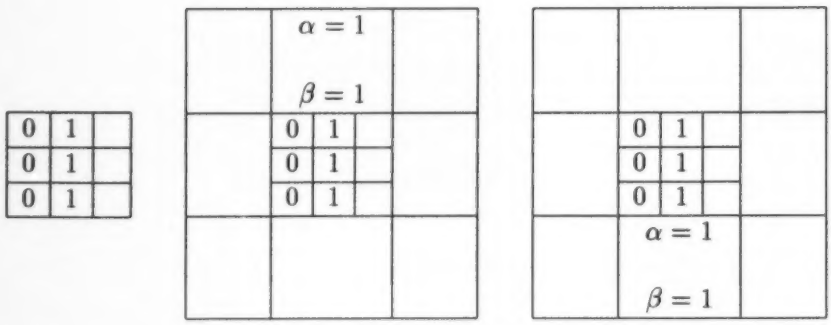


Figure 24: Example templates (a) a 3×3 template, (b) and (c) are variable resolution templates.

The binary variable α is used to determine if the character has been sufficiently thinned in this region. Care needs to be taken to prevent shape distortion once a peripheral region is reduced to almost a line. This is done using variable β which records some special pattern of the pixel arrangement in a 3×3 peripheral region. For example, a 3×3 region with three "1" pixels randomly arranged should be distinguished from three "1" pixels forming a straight line in this region. In this work β is used to represent vertical, horizontal and diagonal patterns. Before deleting a pixel, the α and β values in the periphery are checked to avoid shape distortion. For example, if the pixels immediately above and below the contour pixel are both object pixels (with value "1"), as shown in Figure 24(a), then a set of larger VR templates are used. Figures 24(b) and 24(c) give examples of such templates. Here the value "1" for β represents a vertical edge. Other large templates are similar.

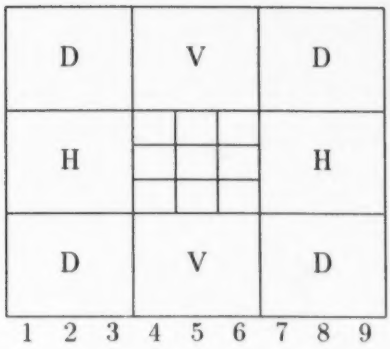


Figure 25: A variable resolution template

These larger templates specify that the character possibly has a long vertical stroke. A contour pixel should NOT be deleted if its neighborhood matches any of these templates.

For any peripheral cell the number of meaningful values of the pair (α, β) is 3. When $\alpha = 0$, β is unimportant. When $\alpha = 1$ only one value of β is checked, depending on the position of the peripheral cell with respect to the central template. This point needs some further clarification. Consider Figure 25 below. For the cells marked V it is required to check if $\beta = x$ vs. $\beta \neq x$, where x is the value of β representing a vertical edge. Similarly, for the cells marked H a test for horizontal edges, and for the corners marked D a check for diagonal edges oriented towards the center is done. This implies that for any outside cell the tests for α and β are: (i) $\alpha = 0$ or $(\alpha = 1, \beta \neq x)$, and (ii) $(\alpha = 1, \beta = x)$. The binary variable γ for each peripheral cell is defined as follows: $\gamma = 0$ if (i) above is true, and $\gamma = 1$ if (ii) holds. Note that at most two of the peripheral cells are considered,

Figure 26: Character "B" and some thinning results using the proposed algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 5, (d) neighborhood of pixel "T", (e) the final skeleton

depending on the conditions prevailing in the inner window. That is, the γ values are generated for at most two peripheral regions.

Very few large templates are needed and the peripheral part of a template can be stored and tested separately from the central part. Figure 26 demonstrates the use of the proposed method. Consider again the input character "B" from Figure 22. Figure 26(b) gives the neighborhood of pixel "Z". Figure 26(c) shows the result of iteration 5 with pixel (18,7) marked "T". The neighborhood of pixel "T" is shown in Figure 26(d). The rules described above can distinguish this neighborhood with that of pixel "Z". Figure 26(e) gives the final result, which looks more like the letter "B" rather

Figure 27: Character "T" and some thinning results using the proposed algorithm (a) input, (b) neighborhood of pixel "Z", (c) after iteration 3, (d) neighborhood of pixel "H", (e) the final skeleton

than numeral "8" since the straight vertical stroke is preserved.

Horizontal and diagonal edge preservation play an equally important role in character recognition. Consider, for example, the characters "T" and "Y". Unless the horizontal stroke on a "T" is retained it may look like a "Y" with a short top part. Figure 27 shows how a VR approach preserves the horizontal part on a "T".

The algorithm used to decide deletion of a pixel keeps a contour connected. This has been proved in detail by Guo and Hall [19]. The VR thinning algorithm, in addition, retains certain pixels if some conditions hold in the periphery. Since a connected outline already exists before the VR method is applied, all that is required to be shown is: peripheral tests do not generate isolated object pixels. To prove this, consider a pixel that is not

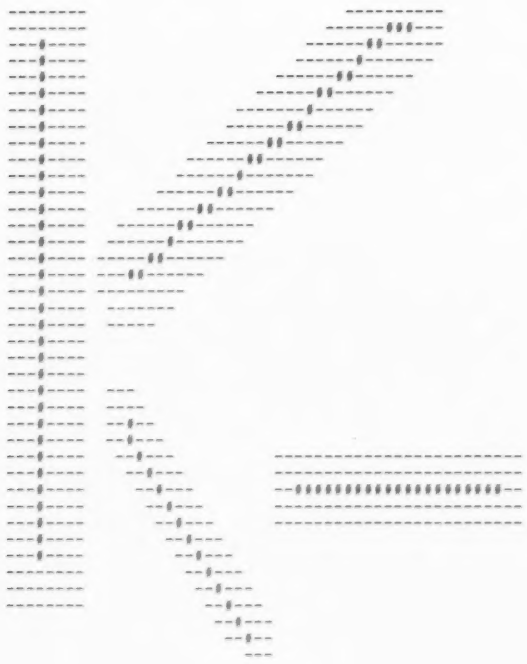


Figure 28: Thinning results of the proposed algorithm for lines of various orientations

deleted because it is part of a vertical stroke, (as in Figure 26(c)). The pixels above and below this pixel will not be deleted because they are either part of the vertical stroke, or part of the periphery which is thinned to almost a line. This method starts retaining the pixels only when the periphery is sufficiently thin. Therefore the peripheral cells are connected by Guo's proof. Also the pixels retained are connected to the peripheral outline by the argument above. Thus the pixels that are not deleted for preserving a vertical stroke do not generate any disconnected contour.

Similar reasoning can be used to show that pixels retained in order to maintain horizontal and diagonal edges keep the resulting contour connected. Experimentally, a situation where the algorithm generated disjoint regions was not encountered. Figure 28 shows the results for lines of various orientations, demonstrating the connectedness property.

5.3 Complexity analysis of sequential method

Over the years different sets of templates and various functions and tests have been proposed for character thinning. The (time and space) complexities of these methods are difficult to compare because of the different ways in which algorithms are implemented. A simple way to directly compare the complexity of thinning algorithms is to convert all the rules (templates, functions, tests) into a large look-up table. If a template of size $n \times n$ is used, $n^2 - 1$ pixels are involved, thus a table of 2^{n^2-1} entries can be used. The table entries are indexed by the values of the neighborhood pixels, and the content of each entry should be either 0 ("do not delete") or 1 ("delete"). The table lookup operation takes one step, so the complexity of the algorithm is completely converted to space complexity, which provides a ground for comparison among algorithms. It is obvious that the size of the table grows rapidly as n increases.

The VR algorithm involves two stages:

(i) Checking the high resolution central window to determine if pixel needs to be deleted.

(ii) Looking at the γ value in the peripheral cells to ensure that deleting

center pixel does not distort the shape of the character being thinned.

A pixel is deleted only if (i) holds, and then (ii) is true. Thus the worst case complexity of the VR algorithm equals the complexity of stage (i) plus the complexity of stage (ii).

Each peripheral cell covers a 3×3 region. Hence the maximum number of possible templates for a peripheral cell is 2^8. The central window can have at most 2^8 combinations. When the high resolution center suggests that a pixel be deleted, at most two peripheral cells need to be checked to confirm the deletion. For this situation the total number of computations is 3×2^8. When the center test does not require deletion there is no need to perform step (ii) above. In this case the computation of γ values is not required, hence the number of operations is at most 2^8. To compute the average number of operations per pixel it is necessary to find out what proportion of pixels in a given array are candidates for deletion. If a fraction p (between 0 and 1) of pixels are considered for deletion then average cost equals $(3p + (1-p))2^8 = (1+2p)2^8$. The proportion p depends on the character being thinned and also on the size of the array in which the digital shape is stored. For example, if the character 'T' is stored in a 10×10 array the number of pixels considered for deletion in a given iteration is at most 20 (one vertical and one horizontal stroke). In this case $p = \frac{20}{100} = 0.2$, and the average complexity of the VR scheme is $(1 + 2 \times 0.2)2^8 = 1.4 \times 2^8$. Similar analysis can be done for other characters and array sizes. For the experiments reported in this thesis a 30×25 array was used. Most characters had 4 prominent strokes or less. Thus for the examples considered p was less than or equal to $\frac{4}{25}$, and the

average complexity was at most $(1 + 2 \times \frac{4}{25})2^8 = 1.32 \times 2^8$. That is, on the average our method needs less than 1.5 times the amount of computation required by procedures using 3×3 windows, for most practical situations.

The analysis given above considers sequential implementation of the algorithm. It is important to note that our method is inherently parallel, and can be very efficiently programmed on parallel computers.

5.4 Experimental results

The proposed algorithm was tested on all the letters of the English alphabet. The experiments with letters demonstrate preservation of vertical, horizontal, and diagonal segments, as well as combinations of them. First the performance of the method on artificially generated binary characters is shown. Then experiments with real images are presented.

In all of the following figures the symbol "–" indicates an object pixel which has been deleted and "#" represents a skeleton pixel. Each figure contains four parts:

(a) input binary image,

(b) final skeleton generated by Holt's method [20] which uses 4×4 templates,

(c) final skeleton generated by Guo's method [19] which uses 3×3 templates,

(d) improved skeleton generated by the variable resolution method.

Figures 29– 30 present the results on characters "D", "P", and "R". For these characters the VR algorithm preserves the vertical edges, which the

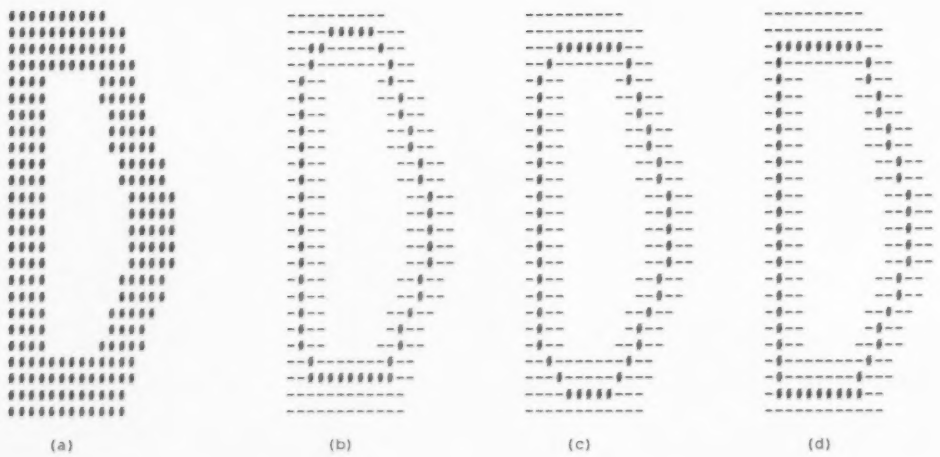


Figure 29: Character "D" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

other two methods fail to do. For the character "D" it can be seen that the previous methods fail to retain the vertical stroke, which makes it difficult to differentiate the resulting outline from "O". The output of the VR procedure does retain the vertical edge on the left, hence the skeleton generated can be clearly distinguished from an "O". A similar phenomenon may be observed in Figure 30. Here the other methods fail to retain the junction between the vertical and horizontal edges in the middle part. This makes the resulting outline appear like "A". Part (d) of the figure shows that the VR technique does not distort the shape.

Examples of horizontal edge preservation are shown in Figures 31 and 32. They illustrate how the edges of characters I and T are better preserved by the variable-resolution scheme.

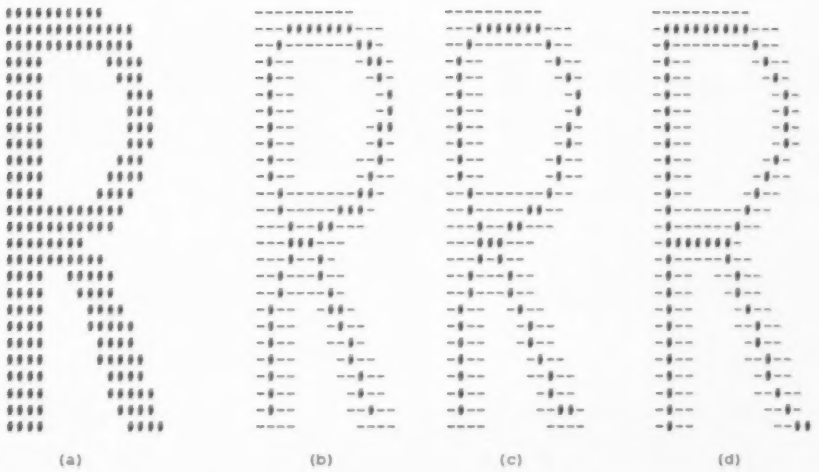


Figure 30: Character "R" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

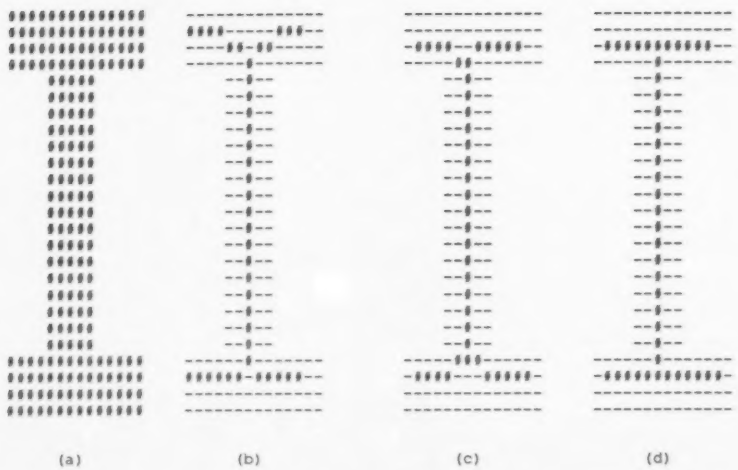


Figure 31: Character "I" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

The distortion of the horizontal edges for the character "I" can cause it to be misclassified as an "X", especially when the vertical stroke is short. Similar problems may arise when the T-junction is not preserved in Figure 32. This can result in the character "T" being interpreted as "Y" by mistake. Such errors are unlikely to occur in a VR scheme, as evident from the results in part (d) of the two examples.

For some other characters such as N and Z, it is important that diagonal edges be preserved. Figures 33– 34 exhibit this capability of the proposed algorithm. The performance of variable-resolution thinning can be observed to be much better for the characters "N" and "Z". The outputs in (b) and (c) of Figure 33 do not seem to resemble "N" as much as in part (d). The distortions caused by Guo's and Holt's approaches (Figure 34) make the

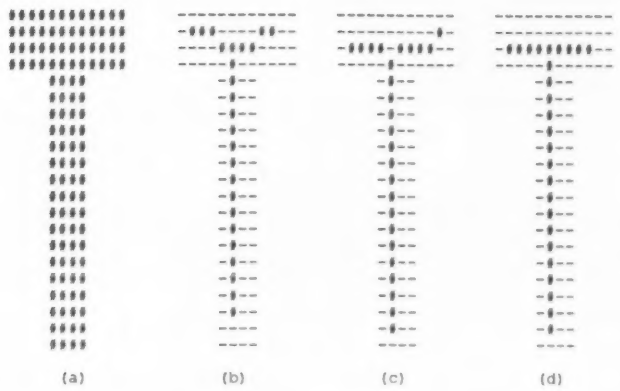


Figure 32: Character "T" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

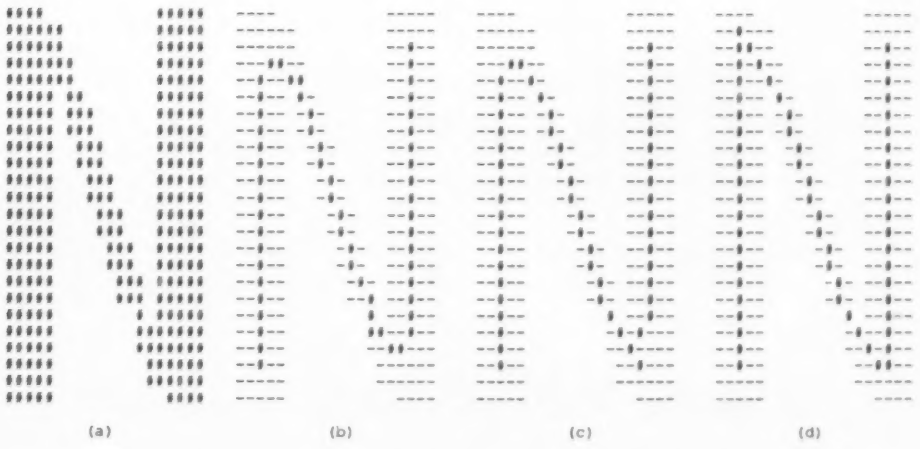


Figure 33: Character "N" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

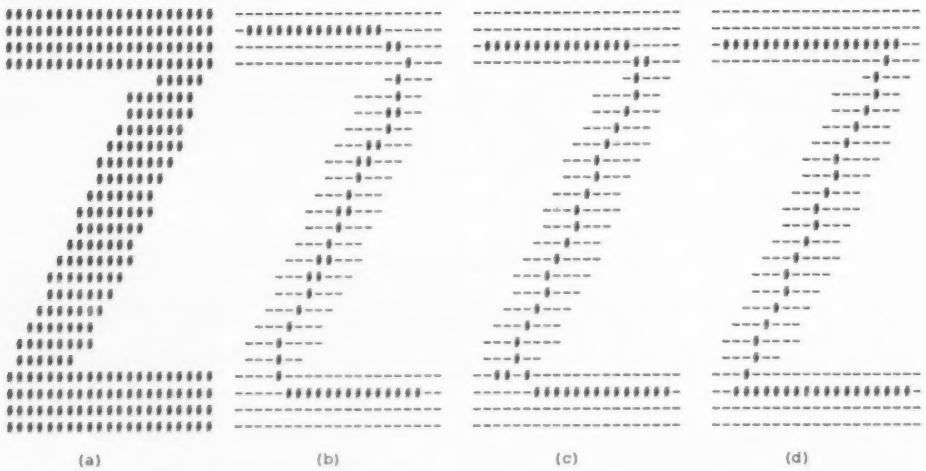


Figure 34: Character "Z" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

resulting skeleton appear like the digit "2". The proposed method, however, preserves the corners sufficiently to obtain an almost perfect "Z".

Several letters, such as E and H, are formed by a combination of horizontal and vertical edges. The output for these characters is shown in Figures 35 and 36. It is easy to recognize "E" as "ϵ", based on the output in Figure 35(b) and (c). Also "H" can be wrongly classified as "X" (Figure 36(b) and (c)). Such misclassification will not occur if the VR approach is used (part (d) of Figures 35 and 36).

A comparison of results for various other letters which do not belong to any of the above categories is shown in Figures 37– 38. Some more skeletons generated by the Guo method, the Holt method, and the VR algorithm are

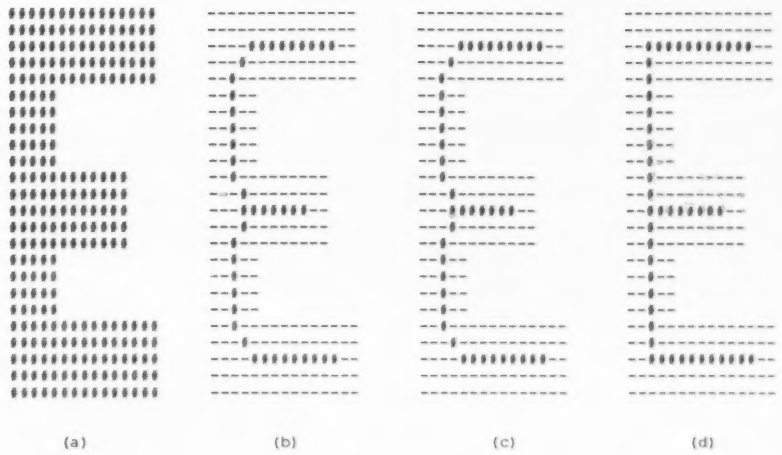


Figure 35: Character "E" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

presented in Appendix B. The present implementation does not attempt to preserve circular or curved edges. This topic is left for future research. It can be observed that the special appendages that are crucial to recognizing the characters "G" and "Q" are fairly well preserved by the VR approach (as well as by Guo's algorithm). Holt's method deletes some special features, making a "G" look like a "C".

The following figures demonstrate the performance of the proposed algorithm on real images of printed text. Figure 39 shows the real image. The image is converted to an integer array and thresholded. Figure 40 shows the image after thresholding. Figures 41– 42 show the thinned outlines obtained by Guo's and Holt's methods respectively. Finally, Figure 43 gives the output

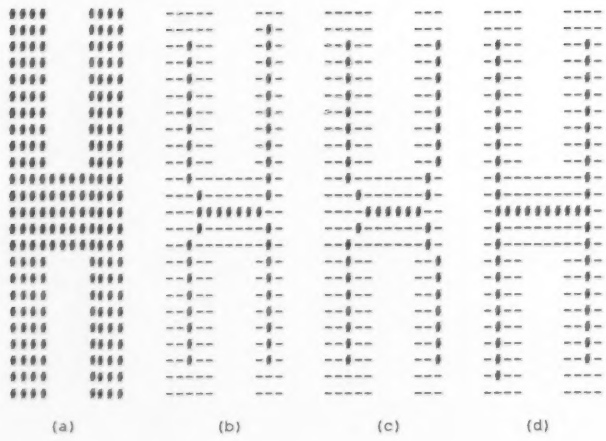


Figure 36: Character "H" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

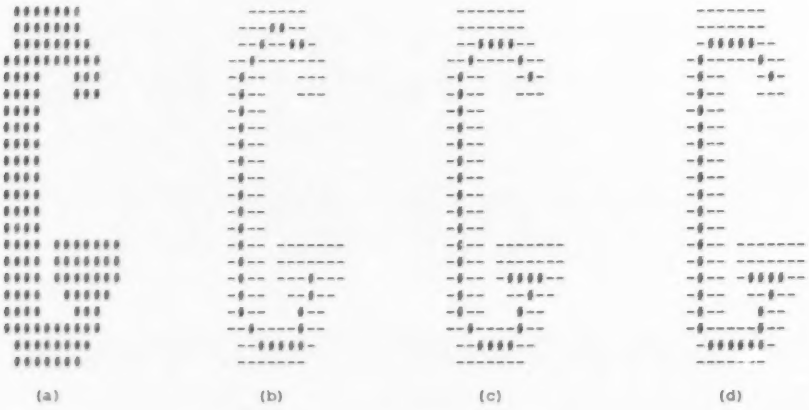


Figure 37: Character "G" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

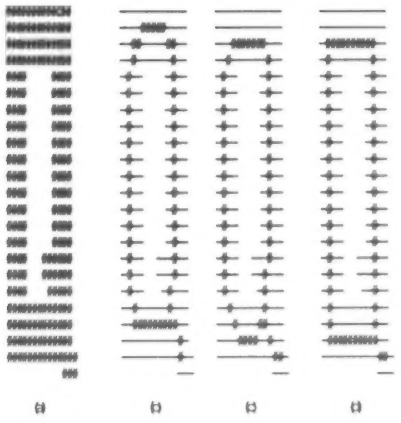


Figure 38: Character "Q" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

of the VR approach. Note that the outline of the VR algorithm is superior to those obtained by other techniques. Also the method preserves the shapes of the characters "R", "E", "P", and "L" much better than both Guo's and Holt's schemes.

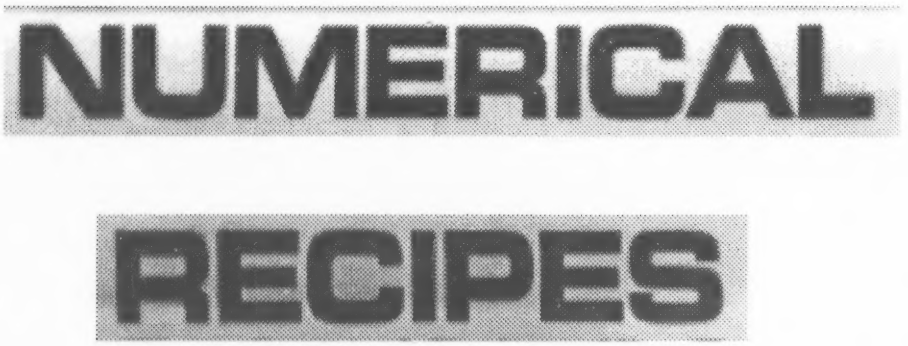


Figure 39: Input Gray Scale Image

Figure 40: Input Gray Scale Image After Thresholding

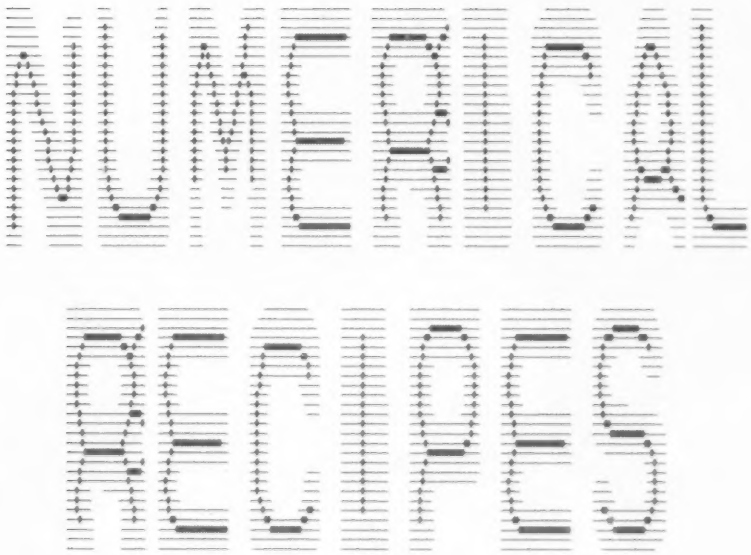


Figure 41: Output of Guo's Thinning Algorithm

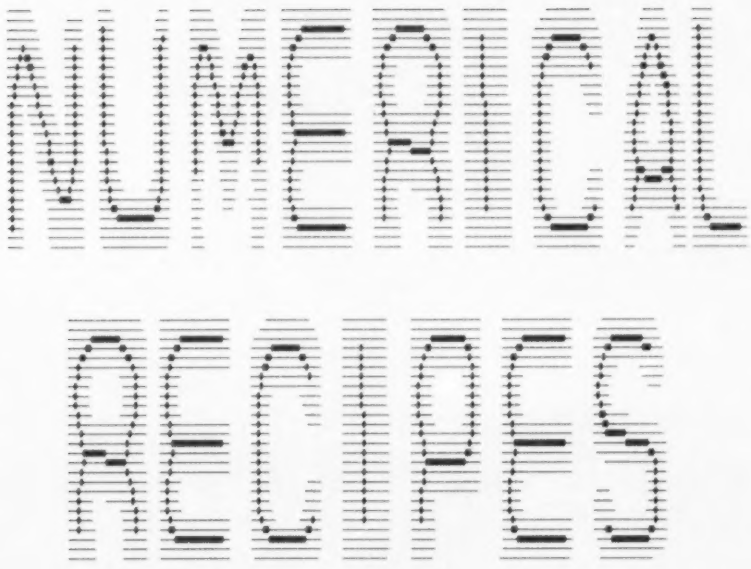


Figure 42: Output of Holt's Thinning Algorithm

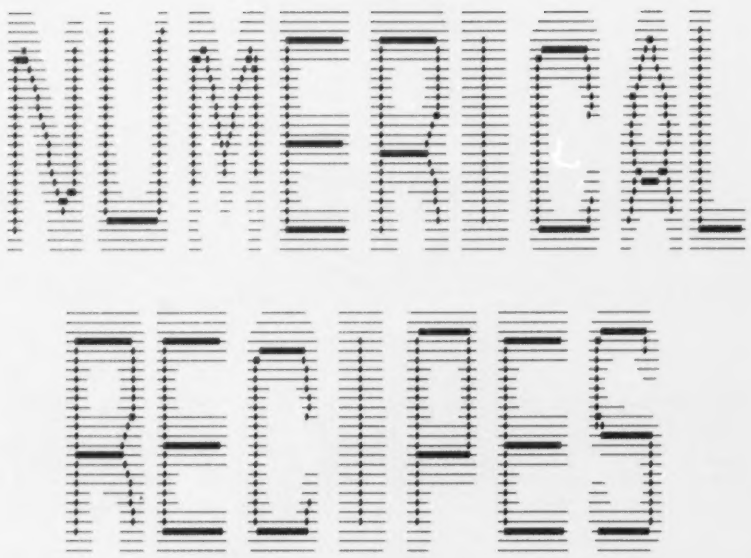


Figure 43: Output of proposed Thinning Algorithm

Chapter 6

Conclusion

6.1 Contribution

In this thesis, edge and corner preserving schemes for thinning and boundary detection were presented, using a new variable-resolution approach. The method takes a "rough" view of the peripheral region outside of the central neighborhood when certain conditions exist in the inner window.

Normally, in existing boundary extraction algorithms the edge detection is performed using the gradient operators, compass operators or using the differentiation method. Though this stage is a parallel process, the subsequent edge linking is carried out as a sequential process. The VR boundary extraction and character thinning algorithms are inherently parallel in nature and are implementable on existing parallel computers. When programmed on a serial computer, the average complexity of the proposed scheme is only a few times higher than methods using small uniform resolution windows.

Extensive experiments demonstrate that the VR scheme performs much better on English letters compared to the best known existing techniques. The VR thinning algorithm is able to preserve the shape of the corners and the vertical and horizontal strokes thereby generating better skeletons. Currently this algorithm works for letters of English language. Similarly, the VR method was able to extract better contours. The scheme is robust in nature and performs well even when images are noisy.

6.2 Directions for Future Work

A gross test of boundary accuracy is whether the boundary succeeds in generally delineating the object. In some cases, the boundary extraction algorithms fail to detect the accurate boundary of the object in the image. Similarly the accuracy of the skeletons generated by the thinning algorithms can only be compared visually. No quantitative measurement for evaluating the algorithms exist in the literature.

However, if such a quantitative criterion can be derived then a true comparison of the techniques, human independent, can be achieved. For boundary extraction, one method to test the accuracy of the algorithm is to store a model of the object and then test the accuracy of the boundary with respect to the model. The accuracy test that can be used here is to measure the distance of the located boundary elements from the model and then apply a statistical analysis to find the mean distance and maximum distance from the ideal boundary. The performance of the method would then be inversely proportional to the value of these two parameters. Though this approach of

testing the robustness of the algorithm is sound, it depends entirely on the edge detector used in the preliminary stage of boundary extraction.

Also, it would be worth while to implement these algorithms on the existing multiprocessor architectures and compare the execution time, speed-up obtained, and the quality of the resultant output with the other methods.

Bibliography

[1] C. Arcelli and G.S. Anuiti Di Baja. A width-independent fast thinning algorithm. *IEEE Trans. PAMI*, 7(4):463–474, 1985.

[2] B. P. Ashkar and J. W. Modestino. The contour extraction problem with biomedical applications. *Computer Graphics and Image Processing*, 7:331–355, 1978.

[3] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[4] A. Basu and X. Li. Variable resolution vision. *Technical Report TR 90-14, Dept. of Computing Science, University of Alberta*, 1990.

[5] R. Bellman and S. Dreyfus. *Applied Dynamic Programming.* Princeton Univ. Press, Princeton, NJ, 1962.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. PAMI*, 8(6):679–698, 1986.

[7] B. D. Chen and P. Siy. Forward/backward contour tracing with feedback. *IEEE Trans. PAMI*, 9(3):438–446, 1987.

[8] Y. Chen and W. Hsu. A modified fast parallel thinning algorithms for digital patterns. *Pattern Recognition Letters*, 7(2):99–106, 1988.

[9] Y. P. Chien and K. S. Fu. A decision function method for boundary detection. *Computer Graphics and Image Processing*, 3:125–140, 1974.

[10] R.T. Chin, H-K Wan, D.L. Stover, and R.D. Iverson. A one-pass thinning algorithm and its parallel implementation. *Computer Vision, Graphics and Image processing*, 40:30–40, 1987.

[11] J. J. Clark. Singularity theory and phantom edges in scale space. *IEEE Trans. PAMI*, 10:720–727, 1988.

[12] L. S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4:248–270, 1975.

[13] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, pages 11–15, 1972.

[14] C. Dyer and A. Rosenfeld. Thinning algorithm for gray-scale pictures. *IEEE Trans. on PAMI*, 1:88–89, 1979.

[15] W. Frei and C. C. Chen. Fast boundary detection: A generalization and a new algorithm. *IEEE Trans. Computers*, 26(10):988–998, 1977.

[16] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, 1992.

[17] V. Govindan and A. Shivaprasad. A pattern adaptive thinning algorithm. *Pattern Recognition*, 20(6):623–637, 1987.

[18] P. Grattoni, F. Pollastri, and A. Premoli. A contour detection algorithm based on the minimum radial inertia. *Computer Vision, Graphics, and Image Processing*, 43:22–36, 1988.

[19] Z. Guo and R. Hall. Parallel thinning with two subiteration algorithms. *Comm. ACM*, 32(3):359–373, 1989.

[20] C. Holt, A. Stewart, M. Clint, and R. Perrott. An improved parallel thinning algorithm. *Comm. ACM*, 30(2):156–160, 1987.

[21] P. V. C. Hough. Methods and means of recognizing complex patterns. US Patent 3069654, 1962.

[22] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ 07632, 1989.

[23] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM Journal Res. Dev.*, 13:675–685, 1969.

[24] C. Kimmie, D.H. Ballard, and J. Sklansky. Finding circles by an array of accumulators. *Comm. ACM*, 18:120–122, 1975.

[25] M. Kunt. Edge detection: A tutorial review. In *Proc. ICASSP*, pages 1172–1175, 1982.

[26] P. Kwok. A thinning algorithm by contour generation. *Comm. ACM*, 31:1314–1324, 1988.

[27] V. Lacroix. A three-module strategy for edge detection. *IEEE Trans. PAMI*, 10(6):803–810, 1988.

[28] S. Levialdi. Edge extraction techniques. INRIA-CREST course on Computer Vision, 1982.

[29] S. Licardie. Survey of edge detection techniques. Project Report for CMPUT509, Department of Computing Science, Univ. of Alberta, 1992.

[30] H. K. Liu. Two and three dimensional boundary detection. *Computer Graphics and Image Processing*, 6:123–134, 1977.

[31] D. Marr. *Vision.* W.H. Freeman: San Francisco, 1982.

[32] D. Marr and E. Hildreth. Theory of edge detection. *Proc. R. Soc. Lond. B*, 207:187–217, 1980.

[33] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1:169–182, 1972.

[34] A. Martelli. An application of heuristic search methods to edge and contour detection. *Comm. ACM*, 19(2):73–83, 1976.

[35] U. Montanari. On the optimal detection of curves in noisy pictures. *Comm. ACM*, 14:335–345, 1971.

[36] T. Pavlidis. A flexible parallel thinning algorithm. *Proc. Conf. Pattern Recognition Image Processing*, pages 162–167, 1981.

[37] T. Peli and D. Malah. A study of edge detection algorithms. *Computer Vision, Graphics, and Image Processing*, 20:1–21, 1982.

[38] R. Plamondon and C.Y. Suen. On the definition of reference skeletons for comparing thinning algorithms. *Proc. Vision Interface '88*, pages 70–75, 1988.

[39] A. Premoli, P. Grattoni, and F. Pollastri. A non-sequential contour detection with a priori knowledge. *Pattern Recognition Letters*, 9:45–51, 1989.

[40] J. M. S. Prewitt. Object enhancement and extraction. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*. Academic Press, New York, 1970.

[41] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, New York, 1976.

[42] J. Rovamo and V. Virsu. An estimation and application of the human cortical magnification factor. *Experimental Brain Research*, 37:495–510, 1979.

[43] G. Sandini and V. Tagliasco. An anthropomorphic retina-like sensor for scene analysis. *Computer Graphics and Image Processing*, 14(3):365–372, 1980.

[44] E. L. Schwartz. Computational anatomy and functional architecture of striate cortex: A spatial-mapping approach to perceptual coding. *Vision Research*, 20:645–670, 1980.

[45] M. Shah, A. Sood, and R. Jain. Pulse and staircase edge models. *Computer Vision, Graphics, and Image Processing*, 34:321–341, 1986.

[46] H. Tamura. A comparison of line thinning algorithms from digital geometry view point. *Proc. 4th Int. Joint Conf. Patt. Rec.*, pages 715–719, 1978.

[47] V. Torre and T. A. Poggio. On edge detection. *IEEE Trans. PAMI*, 8(2):147–163, 1986.

[48] V. Virsu and J. Rovamo. Visual resolution, contrast sensitivity, and the cortical magnification factor. *Experimental Brain Research*, 37:475–494, 1979.

[49] H. Wechsler and J. Sklansky. Finding the rib cage in chest radiographs. *Pattern Recognition*, 9:21–30, 1977.

[50] A. Wigena and X. Li. Thinning algorithms and criteria. *Proc. Canadian Conf. Electrical and Computer Engineering*, pages 765–770, 1989.

[51] D. J. Williams and M. Shah. Edge contours using multiple scales. *Computer Vision, Graphics, and Image Processing*, 51:256–274, 1990.

[52] A. Witkin. Scale-space filtering. In *Int'l Joint Conf. AI*, pages 1019–1021, 1983.

[53] Y. Yeshurun and E.L. Schwartz. Shape description with a space varying sensor: Algorithms for scan-path, fusion, and convergence over multiple scans. *IEEE Trans. PAMI*, 11:1217–1222, 1989.

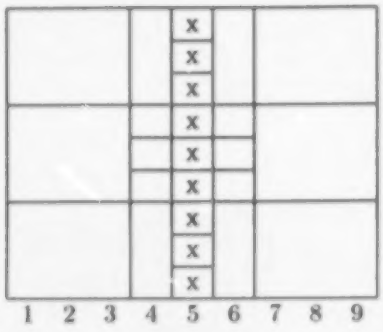
[54] S. Yu and W. Tsai. A new thinning algorithm for gray-scale images by the relaxation technique. *Pattern Recognition*, 23:1067–1076, 1990.

[55] K. Sh. Zigangirov. Some sequential decoding procedures. *Probl. Peredachi Inf.*, 2:13–25, 1966.

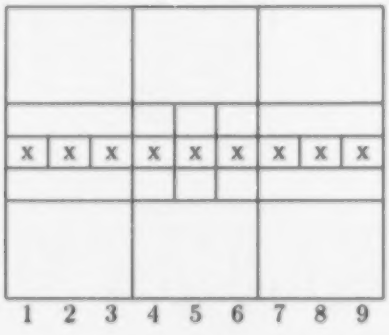
Appendix A

Masks and Rule Windows for Boundary Detection

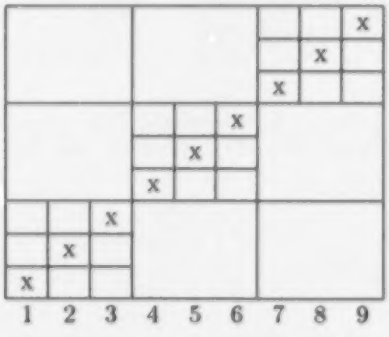
1. Mask for Vertical Edge.



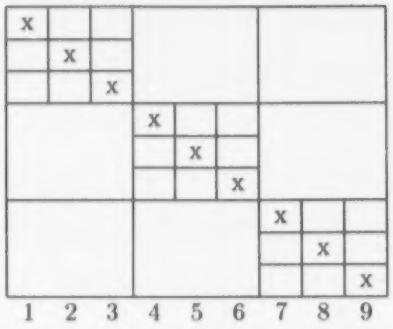
2. Mask for Horizontal Edge



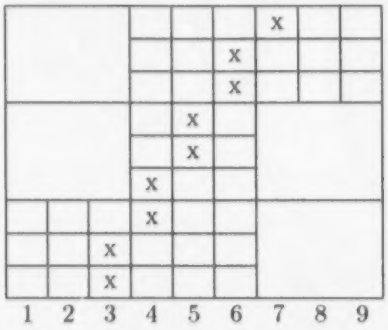
3. Mask for Various Rotated Positions: Position 45 degrees



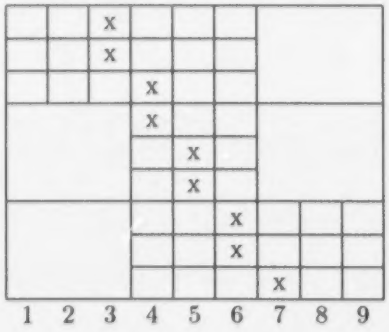
4. Mask for a Rotation of 135 degrees.



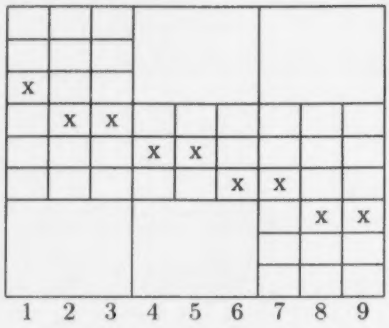
5. Rotation of 67.5 degrees



6. Rotation Mask for 112.5 degrees



7. Rotation Mask for 157.5 degrees



Appendix B

Some more results on Character Thinning

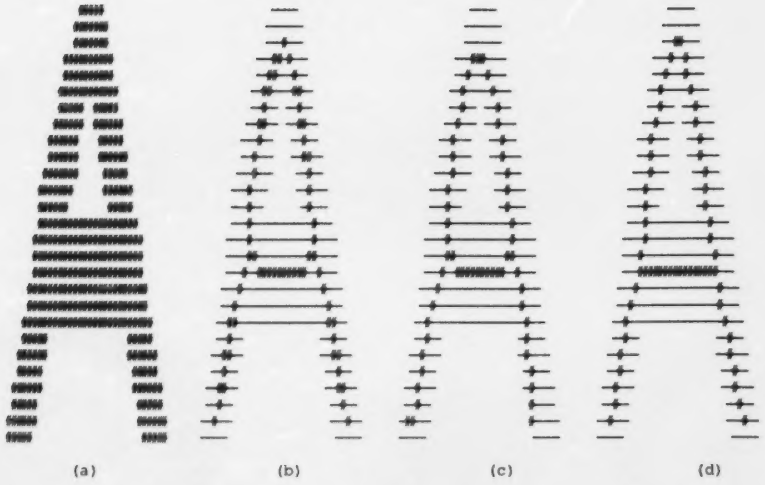


Figure 44: Character "A" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

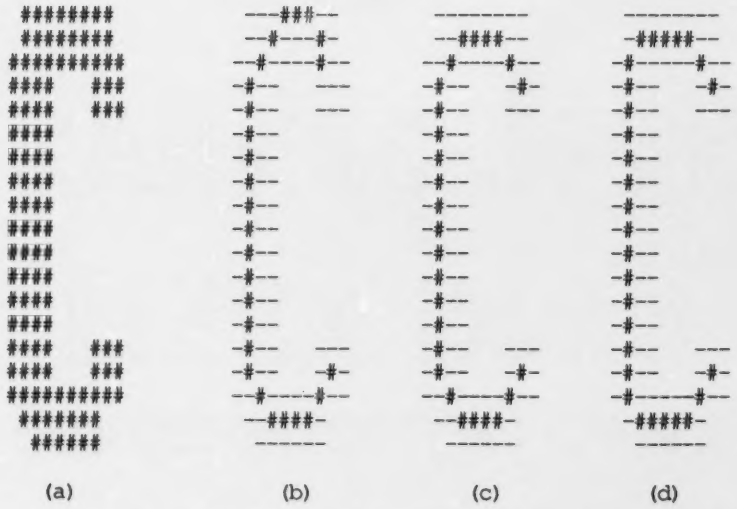


Figure 45: Character "C" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

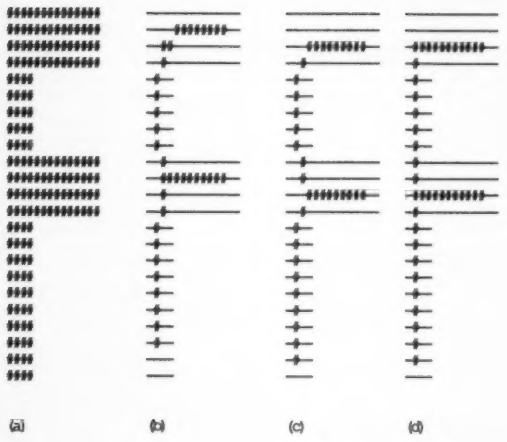


Figure 46: Character "F" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

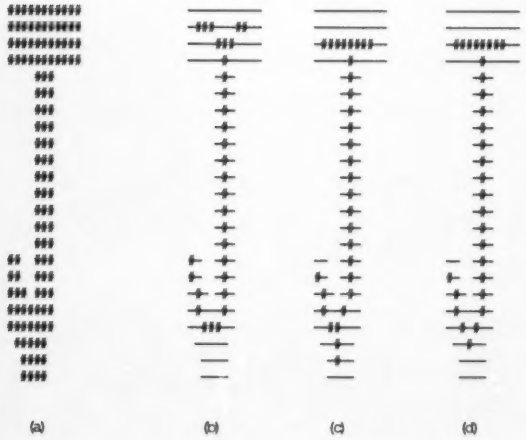


Figure 47: Character "J" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

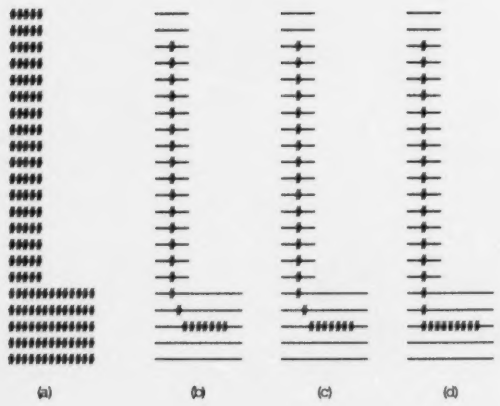


Figure 48: Character "L" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

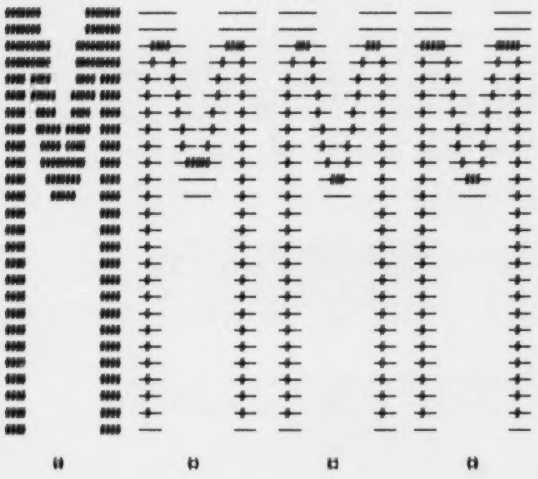


Figure 49: Character "M" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

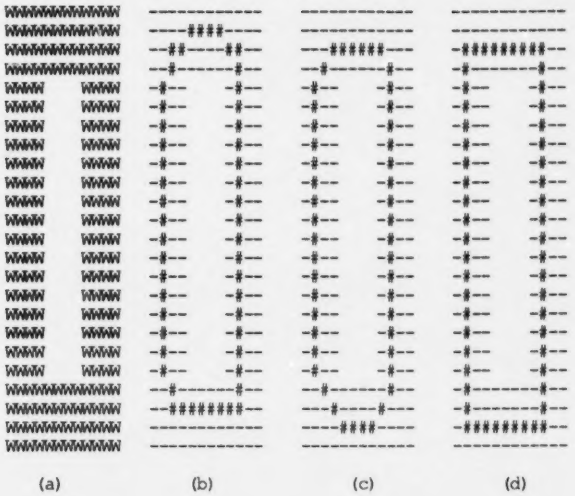


Figure 50: Character "O" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

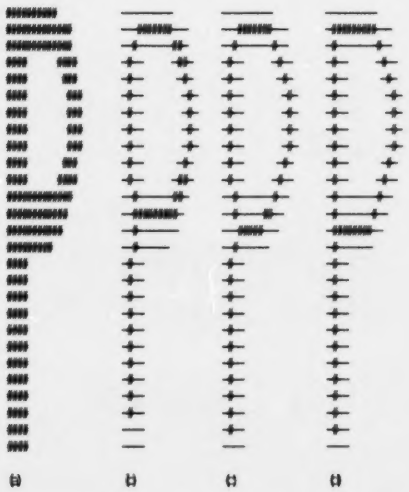


Figure 51: Character "P" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

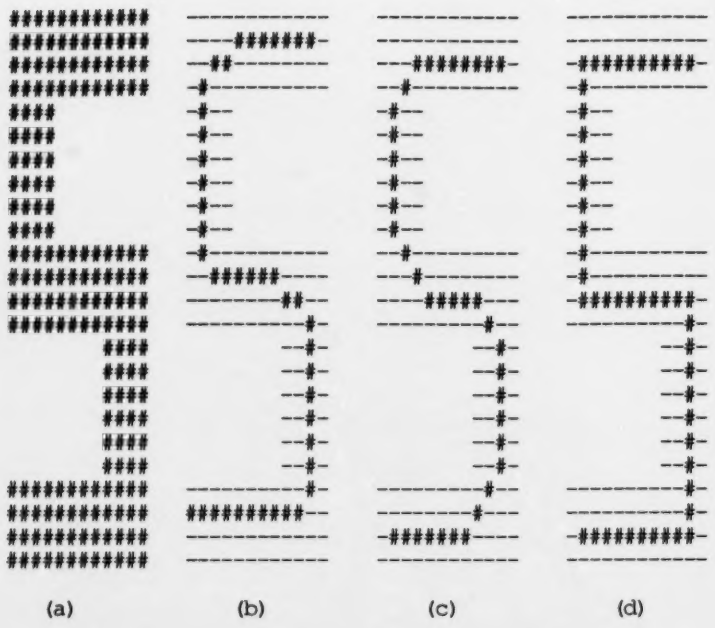


Figure 52: Character "S" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

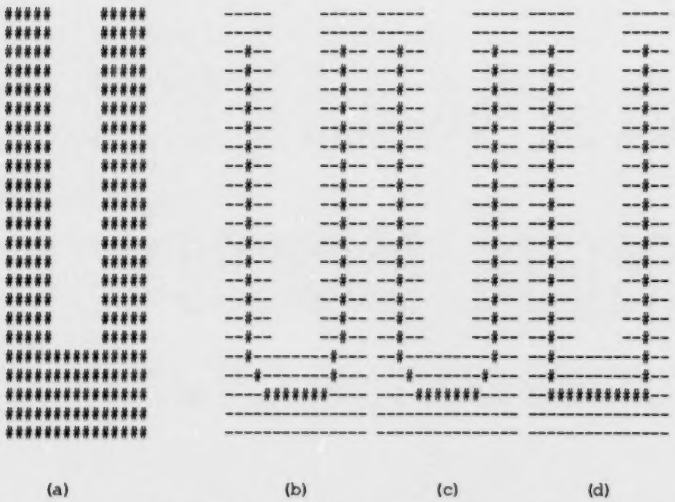


Figure 53: Character "U" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

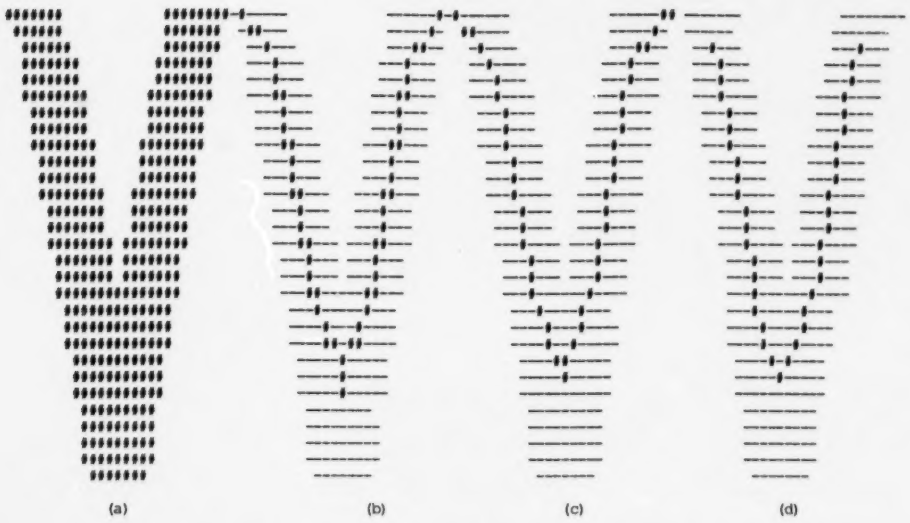


Figure 54: Character "V" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

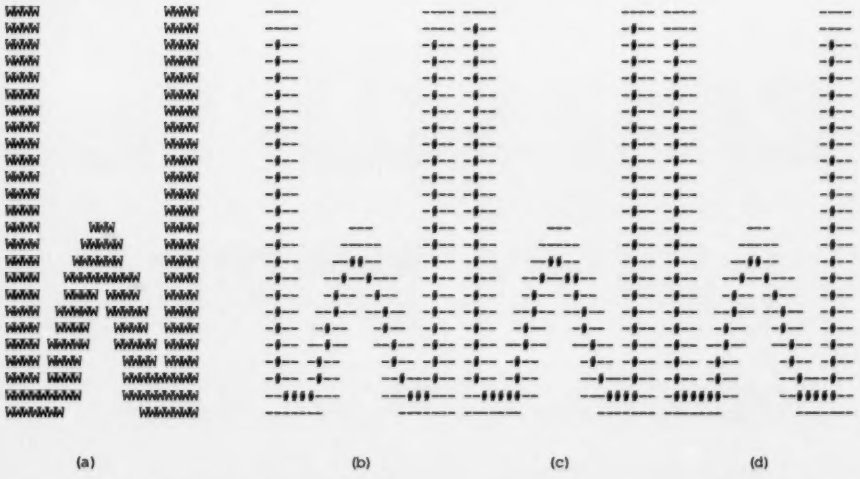


Figure 55: Character "W" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

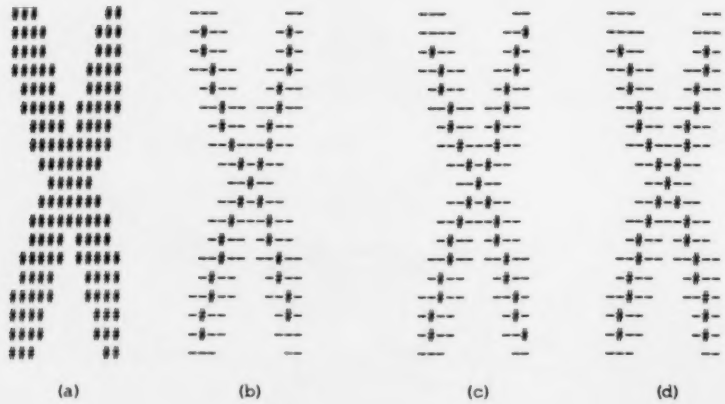


Figure 56: Character "X" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method

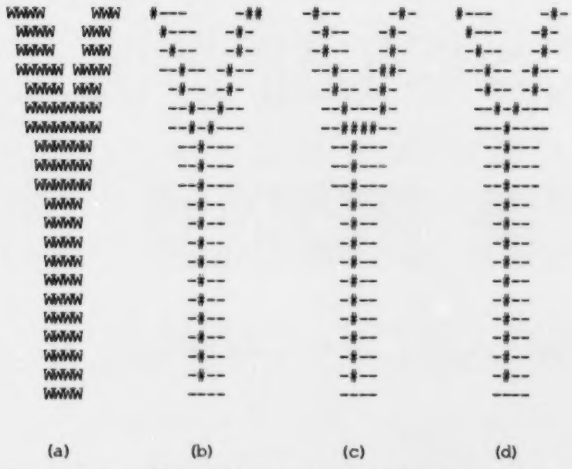


Figure 57: Character "Y" and thinning results of three algorithms (a) input, (b) Holt's method, (c) Guo's method, (d) proposed method